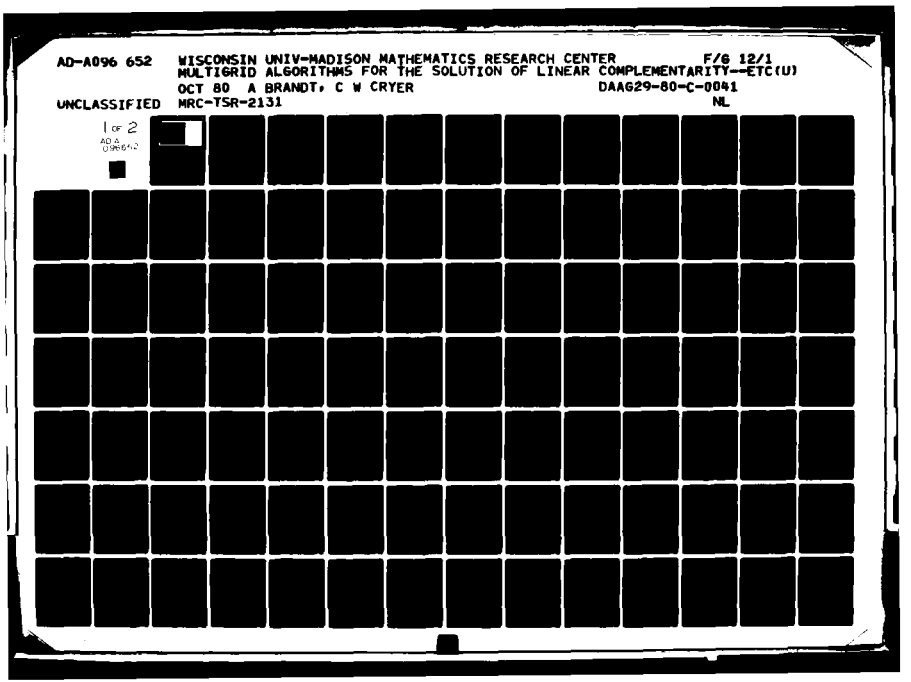

AD-A096 652
WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY--ETC(U)
OCT 80 A BRANDT, C W CRYER
DAAG29-80-C-0041
UNCLASSIFIED
MRC-TSR-2131
NL
1 OF 2
AD A
096652

MRC Technical Summary Report #2131

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS

Achi Brandt and Colin W. Cryer

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706

October 1980

(Received April 3, 1980)

Approved for public release
Distribution unlimited

Sponsored by

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

National Science Foundation
Washington, D.C. 20550

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS

Achi Brandt*,(1) and Colin W. Cryer**,(2)

Technical Summary Report #2131
October 1980

ABSTRACT

We show that the multigrid algorithms of Brandt can be adapted to solve linear complementarity problems arising from free boundary problems. The multigrid algorithms are significantly faster than previous algorithms. Using the multigrid algorithms, which are simple modifications of multigrid algorithms for equalities, it is possible to solve the difference equations to within truncation error using less work than the equivalent of six Gauss-Seidel sweeps on the finest grid.

AMS (MOS) Subject Classifications: 35J65, 35R35, 65N99, 90C33

Key Words: Multigrid Algorithms, Free Boundary Problems, Linear Complementarity Problems

Work Unit Number 3 (Numerical Analysis and Computer Science)

---

* The Weizmann Institute of Science, Department of Applied Mathematics, Rehovot, Israel.

** Computer Sciences Department and Mathematics Research Center, University of Wisconsin-Madison, Madison, WI 53706.

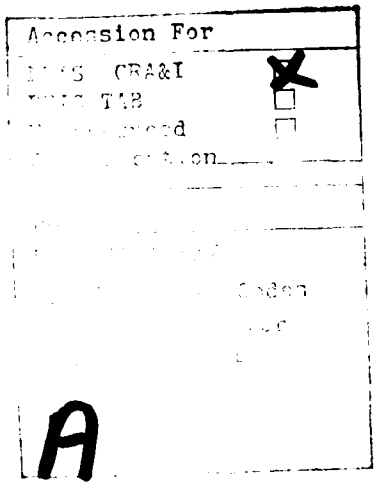
---

(1) Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

(2) Sponsored by the National Science Foundation under Grant No. MCS77-26732 and the United States Army under Contract No. DAAG29-80-C-0041.

SIGNIFICANCE AND EXPLANATION

Several free boundary problems, (including: saturated-unsaturated flow through porous dams; elastic-plastic torsion; and cavitating journal bearings) can be formulated as linear complementarity problems of the following type. Find a non-negative function u which satisfies prescribed boundary conditions on a given domain and which, furthermore, satisfies a linear elliptic equation at each point of the domain where u is greater than zero. We show that the multigrid algorithms of Brandt, (in which solutions are computed on a series of nested grids) which were developed to solve boundary value problems for elliptic partial differential equations, can easily be adapted to handle linear complementarity problems. The resulting algorithms are significantly faster than previous algorithms in which only one grid is used, since the computation time is proportional to the number of gridpoints on the finest grid.



The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the authors of this report.

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS

Achi Brandt*,(1) and Colin W. Cryer**,(2)

1.1 INTRODUCTION.

Several free boundary problems can be reformulated in the form of an (infinite-dimensional) LCP (linear complementarity problem): Given a polygonal domain $\Omega \subset R^n$ with boundary $\partial\Omega$, and given functions f and g, find u (defined on Ω) such that (in an appropriate weak sense)

$$\begin{aligned} &\text{(a)} && \mathcal{L}u(x) \le f(x), && x \in \Omega, \\ &\text{(b)} && u(x) \ge 0, && x \in \Omega, \\ &\text{(c)} && u(x)[\mathcal{L}u(x) - f(x)] = 0, && x \in \Omega, \\ &\text{(d)} && u(x) = g(x), && x \in \partial\Omega. \end{aligned} \tag{1.1}$$

where $\mathcal{L}$ is a given second order elliptic operator. The restriction that Ω is polygonal is not essential, but suffices for our present purposes. We do not write (1.1a) in the more usual form $-\mathcal{L}u(x) + f(x) \ge 0$ because we wish to maintain compatibility with the notation in previous papers by Brandt.

Well-known examples of free boundary problems which can be written in the form (1.1) include porous flow through dams (a recent reference is Baiocchi [1978]), journal bearing lubrication (Cryer [1971a], Cimatti [1977]) and elastic-plastic torsion (Cea, Glowinski, and Nedelec [1974], Lanchon [1974], Cryer [1979]). General references include: Duvaut and Lions [1976]; Glowinski, Lions, and Tremolieres [1976], and Cryer [1977], Glowinski [1978]; Cottle, Giannessi, and Lions [1980]; and Kinderlehrer and Stampacchia [1980].

* The Weizmann Institute of Science, Department of Applied Mathematics, Rehovot, Israel

** Computer Sciences Department and Mathematics Research Center, University of Wisconsin-Madison, Madison, Madison, WI 53706.

(1) Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

(2) Sponsored by the National Science Foundation under Grant No. MCS77-26732 and the United States Army under Contract No. DAAG29-80-C-0041.

If Ω is approximated by a regular grid then the grid can be divided into $N = |G|$ "interior" points G and $|\partial G|$ "boundary" points ∂G. Let the grid size be h. When (1.1) is approximated using finite differences on G, one obtains a (finite-dimensional) LCP:

$$\begin{aligned}
&\text{(a)} && LU(x) \leq f(x), && x \in G, \\
&\text{(b)} && U(x) \geq 0, && x \in G, \\
&\text{(c)} && U(x)[LU(x) - f(x)] = 0, && x \in G, \\
&\text{(d)} && U(x) = g(x), && x \in \partial G,
\end{aligned} \tag{1.2}$$

where $U(x)$ is an approximation to $u(x)$ at the grid points $x \in G \cup \partial G$ and where L is a difference operator which approximates $\mathcal{L}$. The coefficients of L are $0(h^{-2})$.

By multiplying (1.2) by h^2 and eliminating the known values of $U(x)$ on ∂G, the LCP (1.2) may be written in matrix form

$$\begin{aligned}
&\text{(a)} && AU \leq b, \\
&\text{(b)} && U \geq 0, \\
&\text{(c)} && U^T(AU - b) = 0,
\end{aligned} \tag{1.3}$$

where U is the N-vector of values of $U(x)$ on G, and A is an $N \times N$ matrix with coefficients which are $0(1)$. Since we will assume that A is symmetric and negative definite, (1.3) could be brought into the canonical form for an LCP by multiplying (1.3a) by -1.

For example, if $\mathcal{L}$ is the Laplace operator in R^2, then a possible choice for L would be the classical five-point difference operator, in which case A would be a matrix with diagonal elements -4 and off-diagonal elements either 0 or 1.

The general structure of a finite-dimensional LCP is that we have a pair of vector inequalities together with the complementarity condition which states that at every point at least one of the inequalities must in fact be an equality.

There is an extensive literature on the (finite-dimensional) LCP (see Balinski and Cottle [1978]). In particular, if A is negative definite, as we assume, then there exists a unique solution to (1.2) and (1.3).

Since the LCP (1.3) arises from a free boundary problem, the matrix A has special properties which make it possible to use specialized algorithms which are particularly efficient. Such algorithms include projected SOR (Cryer [1971], Glowinski [1971]) the method of Cottle and Sacher [1977], and the modified block SOR (MBSOR) method of Cottle, Golub, and Sacher [1978]; Cryer [1979a] summarizes these algorithms and Cottle [1974] gives numerical comparisons between them.

Recently, it has been found (Brandt [1977], Brandt and Dinar [1979]) that multigrid algorithms are an effective tool for solving linear equations of the form

$$AX = b . \tag{1.4}$$

The basic idea of these multigrid algorithms is to compute on a sequence of nested grids. The computation proceeds on a particular grid until the error becomes smooth and the rate of convergence slows, at which point the computation is transferred to a coarser grid. When the error has been reduced on the coarser grid, the solution on the finer grid is corrected using interpolated values from the coarser grid.

In this paper, we show how the multigrid algorithms FAS and FMG of Brandt can be modified to solve the LCP (1.3). We find that the modified multigrid algorithms are substantially faster than previous algorithms. Indeed, with only minor modifications, the standard multigrid programs solve the LCP with essentially the same efficiency as is attained for linear equations.

The paper is organized as follows. In Section 2, we describe PFAS, the projected full approximation scheme for solving (1.3); PFAS combines the concepts of multigrid algorithms with those of projected SOR. In Section 3, we discuss the implementation of PFAS, and in Section 4, we give numerical results obtained using PFAS. In Section 5, we discuss alternative implementations of PFAS, the last of which leads to substantially improved convergence (we also include several unsuccessful implementations because they are instructive).

In Section 6, we describe results obtained using PFMG, the projected full multigrid algorithm for solving (1.3). The basic idea of PFMG is to compute the initial

approximation on each grid by interpolating an accurate solution on the next coarser grid. Using PFMG we are able to solve the LCP to within truncation error using less work than the equivalent of six Gauss-Seidel sweeps on the finest grid.

Our results are summarized in Section 7 and some possible extensions are mentioned. Finally, listings of the programs are given in the appendices.

2. PFAS (PROJECTED FULL APPROXIMATION SCHEME).

Brandt [1977], and Brandt and Dinar [1979] give a detailed exposition of multigrid methods and their philosophy, and the reader is referred to these papers for background information. The algorithm described below, PFAS, is a modification of the FAS (Full Approximation Scheme) which is considered in Section 5 of Brandt [1977], and Section 2.2 of Brandt and Dinar [1979].

The polygonal domain $\Omega \subset R^n$ is approximated by a sequence of grids

$$G^1 \subset G^2 \subset \ldots \subset G^M \subset R^n ,$$

with corresponding grid sizes

$$h_1 = 2h_2 = 4h_3 = \ldots = 2^{M-1}h_M .$$

Let F^k be the restriction of f to G^k,

$$F^k(x) = f(x), \quad x \in G^k. \tag{2.1}$$

Then, on G^k the difference equations (1.2) approximating (1.1) take the form

$$\begin{aligned}
&\text{(a)} && L^kU^k(x) \le F^k(x), && \text{in } G^k ,\\
&\text{(b)} && U^k(x) \ge 0, && \text{in } G^k ,\\
&\text{(c)} && U^k(x)[L^kU^k(x) - F^k(x)] = 0, && \text{in } G^k ,\\
&\text{(d)} && U^k(x) = g(x), && \text{in } \partial G^k .
\end{aligned} \tag{2.2}$$

Let the points of G^k be ordered: $x_1^k, x_2^k, \ldots, x_{N_k}^k \in G^k$, and let U^k be the vector

$$U^k = \{U_j^k : 1 \le j \le N_k\} \equiv \{U^k(x_j^k) : 1 \le j \le N_k\} .$$

Then, (1.3) takes the form

$$\begin{aligned}
&\text{(a)} && A^kU^k \le b^k ,\\
&\text{(b)} && U^k \ge 0 ,\\
&\text{(c)} && (U^k)^T[A^kU^k - b^k] = 0 .
\end{aligned} \tag{2.3}$$

where

$$A^k = \{a_{ij}^k : 1 \le i, j \le N_k\} , \tag{2.4}$$

is a known sparse symmetric negative definite matrix and $b^k = \{b_j^k\}$ is a known vector with components $b_j^k = h_k^2 F^k(x_j^k)$ (except at points x_j^k adjacent to ∂G^k).

THE PROJECTED GAUSS-SEIDEL ALGORITHM

It is possible to solve the LCP's (2.2) and (2.3) using the projected Gauss-Seidel algorithm which we now describe.

Let $u^{k,0}(x)$ be an approximate solution of (2.2) and (2.3). We compute recursively a sequence of approximations $u^{k,1}(x)$, $u^{k,2}(x),\ldots,$ as follows. Let $u^{k,s-1}(x)$ be given. From (2.2d), the boundary values of $u^{k,s}(x)$ are equal to $g(x)$. The interior values of $u^{k,s}(x)$, which together comprise the vector

$$u^{k,s} = \{u_j^{k,s} : 1 \le j \le N_k\} = \{u^{k,s}(x_j^k) : 1 \le j \le N_k\}\ , \tag{2.5}$$

are obtained, point by point, by first applying the classical Gauss-Seidel method to (2.3) to obtain

$$\begin{aligned} u_j^{k,s-\frac{1}{2}} &= u_j^{k,s-1} + [b_j^k - \sum_{\ell<j} a_{j\ell}^k u_\ell^{k,s} - \sum_{\ell\ge j} a_{j\ell}^k u_\ell^{k,s-1}]/a_{jj}^k\ , \\ &= u_j^{k,s-1} + \tilde{r}_j^{k,s}/a_{jj}^k\ , \quad \text{say,} \end{aligned} \tag{2.6}$$

and then projecting:

$$u_j^{k,s} = \max\{0, u_j^{k,s-\frac{1}{2}}\}\ . \tag{2.7}$$

The process of applying (2.6) and (2.7) for $1 \le j \le N_k$ to obtain $u^{k,s}$ from $u^{k,s-1}$ will be called a G^k projected Gauss-Seidel sweep, or a G^k projected sweep. The quantities $\tilde{r}_j^{k,s}$ will be called the dynamic residuals.

It is known (Cryer [1971], Glowinski [1971]) that $u^{k,s} \to U^k$ as $s \to \infty$.

When implementing the projected Gauss-Seidel method only the latest values of the solution are stored. We will, therefore, often suppress the iteration counter s and denote one projected Gauss-Seidel sweep applied to (2.2) and (2.3) by

$$u^k \leftarrow \text{Projected Gauss-Seidel}\ [u^k : L^k, F^k]\ . \tag{2.8}$$

Similarly,

$$\nabla u^k = u^{k,s} - u^{k,s-1} \tag{2.9}$$

will denote the difference between the latest approximation u^k and its predecessor, while

$$\nabla u^k_{old} = u^{k,s-1} - u^{k,s-2} , \tag{2.10}$$

denotes the previous difference.

ERROR ESTIMATES FOR THE PROJECTED GAUSS-SEIDEL ALGORITHM

When implementing the projected Gauss-Seidel algorithm as part of a multigrid process, it is important to be able to estimate the error. In order to do so, we note that since, by assumption, $-A^k$ is symmetric and positive definite, there exists a coercitivity constant $\alpha_k > 0$ such that

$$w^T(-A^k)w \geq \alpha_k w^T w , \tag{2.11}$$

for all $w \in R^{N_k}$.

Lemma 2.1

Let U^k be the solution of the LCP (2.3), and let $u^k \geq 0$ be an approximate solution. Let

$$r^k = (r^k_j) = b^k - A^k u^k , \tag{2.12}$$

and $r^k_+ = (r^k_{+j})$, where

$$r^k_{+j} = \begin{cases} r^k_j, & \text{if } u^k_j > 0 , \\ \min\{0, r^k_j\}, & \text{if } u^k_j = 0 . \end{cases} \tag{2.13}$$

Then

$$(U^k - u^k)^T(-A^k)(U^k - u^k) \leq (U^k - u^k)^T(-r^k_+) . \tag{2.14}$$

Hence,

$$\|U^k - u^k\|_2 \leq \alpha_k^{-1}\|r^k_+\|_2 . \tag{2.15}$$

Proof: With r^k_+ defined as above, we see that u^k satisfies the LCP:

$$\begin{aligned} &\text{(a)} && A^k u^k \leq b^k - r^k_+ , \\ &\text{(b)} && u^k \geq 0 , \\ &\text{(c)} && (u^k)^T(A^k u^k - b^k + r^k_+) = 0 . \end{aligned} \tag{2.16}$$

Following Falk [1974] we multiply (2.3a) by the non-negative vector $(u^k)^T$ and use the complementarity condition (2.3c) to obtain

$$(u^k - U^k)^T A^k U^k \leq (u^k - U^k)^T b^k \ . \qquad (*)$$

Similarly, multiplying (2.16a) by $(U^k)^T$ we obtain

$$(U^k - u^k)^T A^k u^k \leq (U^k - u^k)^T (b^k - r_+^k) \ . \qquad (**)$$

Adding (*) and (**) and combining terms we obtain (2.14) and hence (2.15). □

Lemma 2.2.

Let U^k be the solution of the LCP (2.3), and let $u^k \geq 0$ be an approximate solution obtained after one or more G^k projected sweeps. Let

$$A^k = (D^k - L^k - P^k) \qquad (2.17)$$

where D^k is diagonal, and L^k and P^k are strictly lower and upper triangular matrices, respectively.

Then u^k satisfies the LCP

$$\begin{aligned} A^k u^k &\leq b^k - P^k \nabla u^k \ , \\ u^k &\geq 0 \ , \\ (u^k)^T (A^k u^k - b^k + P^k \nabla u^k) &= 0 \ . \end{aligned} \qquad (2.18)$$

Hence,

$$\|U^k - u^k\|_2 \leq \alpha_k^{-1} \|P^k\|_2 \|\nabla u^k\|_2 \ . \qquad (2.19)$$

Proof: Consider the projected Gauss-Seidel method defined by (2.6) and (2.7). For each point x_j^k we first compute the dynamic residual $\tilde{r}_j^{k,s}$. The new value of $u_j^{k,s}$ is chosen so as to reduce the residual. Denote the residual at the point x_j^k immediately after step (2.7) by $\hat{r}_j^{k,s}$, so that

$$\hat{r}_j^{k,s} = \tilde{r}_j^{k,s} - a_{jj}^k (u_j^{k,s} - u_j^{k,s-1}) \ . \qquad (2.20)$$

Remembering that A^k is negative definite, and hence $a_{jj}^k < 0$, we see that there are two possibilities:

$$\begin{aligned} &\text{either } u_j^{k,s} > 0 \text{ and } \hat{r}_j^{k,s} = 0 \ , \\ &\text{or } u_j^{k,s} = 0 \text{ and } \hat{r}_j^{k,s} \geq 0 \ . \end{aligned}$$

Thus, dropping the superscript s, and setting $\hat{r}^k = \{\hat{r}_j^k : 1 \leq j \leq N_k\}$,

$$u^k \geq 0 ,$$
$$\hat{r}^k \geq 0 , \tag{2.21}$$
$$(u^k)^T \hat{r}^k = 0 .$$

Let

$$r^k = b^k - A^k u^k .$$

It is readily seen from (2.17) that

$$\begin{aligned} r^k &= \hat{r}^k + P^k (u^{k,s} - u^{k,s-1}) , \\ &= \hat{r}^k + P^k \nabla u^k . \end{aligned} \tag{2.22}$$

Combining (2.21) and (2.22) we obtain (2.18). Comparing (2.16) and (2.18) we see that the arguments which led to (2.15) from (2.16) may be applied to (2.18), with r_+^k replaced by $-P^k \nabla u^k$, to obtain (2.19). □

As Lemmas 2.1 and 2.2 show, we can estimate the error in an approximate solution u^k in terms of the residual r^k or the difference ∇u^k; we will usually use ∇u^k to estimate the error, since this quantity is readily available during a G^k projected sweep.

Remark. The reader may wonder why we bothered to introduce r_+^k in Lemma 2.1, since (2.15) holds with r_+^k replaced by r^k. The reason is that for the LCP (2.3) there may be large positive residuals at points x_j^k where $U^k(x_j^k) = 0$, but this does not mean that the error is large. □

In multigrid algorithms it is necessary to compare norms on different grids. We, therefore, wish to introduce a norm which is not grid dependent. To do so, we proceed as follows.

We first note that, to a good approximation, the coercivity constant α_k for $-A^k$ satisfies

$$\alpha_k \doteq \alpha h^2 ,$$

where α is the smallest eigenvalue of $\mathcal{L}$.

Next, assume that the approximate grid function u^k has been extended to a function $u^k(x)$ on Ω approximating the solution $u(x)$ of (1.1). Then

$$\|u(x) - u^k(x)\|_{2,\Omega} = \left| \int_\Omega |u(x) - u^k(x)|^2 dx \right|^{\frac{1}{2}} ,$$

$$\doteq \left| \sum_{j=1}^{N_k} h_k^n |U_j^k - u_j^k|^2 \right|^{\frac{1}{2}} ,$$

$$= h_k^{\frac{n}{2}} \|U^k - u^k\|_2 ,$$

$$\leq \frac{h_k^{\frac{n}{2}}}{\alpha_k} \|P^k\|_2 \|\nabla u^k\|_2 ,$$

$$\doteq \frac{\|P^k\|_2}{\alpha} h_k^{\frac{n}{2}-2} \|\nabla u^k\|_2 .$$

The norms $\|P^k\|_2$ are essentially independent of k; for example, for the five-point formula, $\|P^k\|_2 \leq 2$. Thus a measure for the error $\|u(x) - u^k(x)\|_{2,\Omega}$ is provided by

$$\|\nabla u^k\|_G \equiv h_k^{\frac{n}{2}-2} \|\nabla u^k\|_2 , \tag{2.23}$$

and this norm will be used in the computations.

PFAS (PROJECTED FULL APPROXIMATION SCHEME).

PFAS (Projected Full Approximation Scheme) obtains an approximation $\bar{u}^M$ to the solution U^M on the finest grid G^M by recursively generating a sequence of approximations $\bar{u}^k$ on the grids G^k.

Each $\bar{u}^k$ is an approximate solution to an LCP of the form (2.2) with F^k replaced by a function $\bar{F}^k$ which is defined later. In general, $\bar{F}^k$ is different from F^k so that $\bar{u}^k$ is not an approximation to U^k. However, $\bar{F}^M = F^M$ and so $\bar{u}^M$ is an approximation to U^M.

We begin by initializing $\bar{u}^M$ to some suitable value. For example, we might set

$$\begin{aligned} \bar{u}^M(x) &= g(x), \quad \text{on} \quad \partial G^M , \\ \bar{u}^M(x) &= 0 \qquad \text{in} \quad G^M . \end{aligned} \tag{2.24}$$

We also set

$$\|\nabla \bar{u}^M\|_G = 10^{30}, \ \epsilon^M = \epsilon , \tag{2.25}$$

(where ϵ is the desired accuracy on the finest grid, and where the astronomical number 10^{30} ensures that at least two G^M projected sweeps are carried out),

$$\bar{F}^M(x) = F^M(x), \quad \text{for} \quad x \in G^M ,$$

and (2.26)

$$\bar{U}^M(x) = U^M(x), \quad \text{for} \quad x \in G^M .$$

We now make a number of G^M projected sweeps,

$$\bar{u}^M \leftarrow \text{Projected Gauss-Seidel} \quad [\bar{u}^M : L^M, \bar{F}^M] . \tag{2.27}$$

After each sweep we test whether

$$\|\nabla \bar{u}^M\|_G \leq \epsilon^M . \tag{2.28}$$

If so, the accuracy criterion is satisfied, and we accept $\bar{u}^M$ as an accurate approximation to $U^M \equiv \bar{U}^M$ on G^M.

It is known that Gauss-Seidel iteration is a smoothing process: the error $\bar{U}^M(x) - \bar{u}^M(x)$ becomes smoother as the number of sweeps increases, while, at the same time, the rate of convergence slows down. We, therefore, carry out only a few G^M projected sweeps, stopping when either (2.28) is satisfied or

$$\|\nabla \bar{u}^M\|_G \geq \eta \|\nabla \bar{u}^M_{old}\|_G . \tag{2.29}$$

Here, η is a fixed parameter; in our work we have taken $\eta = .5$.

Suppose that (2.28) is not satisfied but that (2.29) is satisfied. This means on the one hand that the accuracy of $\bar{u}^M$ must be improved and on the other hand that it is inefficient to continue iterating on G^M. The slow rate of convergence on G^M indicates that the error is smooth, so that the error can be represented satisfactorily to the next coarsest grid, G^{M-1}. We therefore move to G^{M-1}.

Since $\bar{U}^M(x)$ satisfies (2.2), with $k = M$ and $F^M = \bar{F}^M$, the error

$$V^M(x) = \bar{U}^M(x) - \bar{u}^M(x) , \tag{2.30}$$

satisfies the LCP

$$\begin{aligned}
L^M V^M(x) &\geq \bar{r}^M(x), \quad \text{on} \quad G^M , \\
V^M(x) + \bar{u}^M(x) &\geq 0, \quad \text{on} \quad G^M , \\
[V^M(x) + \bar{u}^M(x)][L^M V^M(x) - \bar{r}^M(x)] &= 0, \quad \text{on} \quad G^M , \\
V^M(x) &= 0 , \quad \text{on} \quad \partial G^M ,
\end{aligned} \tag{2.31}$$

where the residual $\bar{r}^M$ is given by

$$\bar{r}^M(x) = \bar{F}^M(x) - L^{M-M}\bar{u}^M(x), \quad x \in G^M . \tag{2.32}$$

As already observed, $V^M(x)$ *is a smooth* function and may, therefore, be accurately represented on G^{M-1}. Furthermore, comparing (2.31) and (1.1) we see that $V^M(x)$ is an approximation to the continuous solution $v(x)$ of the LCP

$$\begin{aligned} \mathcal{L}v(x) &\leq \bar{r}^M(x), \quad x \in \Omega , \\ v(x) + \bar{u}^M(x) &\geq 0, \qquad x \in \Omega , \\ [v(x) + \bar{u}^M(x)][\mathcal{L}v(x) - \bar{r}^M(x)] &= 0 \qquad x \in \Omega , \\ v(x) &= 0, \quad \text{on} \quad \partial\Omega , \end{aligned} \tag{2.33}$$

(where, by abuse of notation, $\bar{r}^M(x)$ and $\bar{u}^M(x)$ are defined on Ω by appropriate interpolation between the values of $\bar{r}^M$ and $\bar{u}^M$ on the gridpoints of G^M). Thus, a good approximation to $V^M(x)$ may be obtained by solving the finite difference approximation to (2.33) on G^{M-1}. That is, $V^M(x)$ is closely approximated on G^{M-1} by the solution $W^{M-1}(x)$ of the LCP,

$$\text{(a)} \qquad L^{M-1}W^{M-1}(x) \leq S_M^{M-1}\bar{r}^M(x), \quad \text{on} \quad G^{M-1} ,$$

$$\text{(b)} \qquad W^{M-1}(x) + I_M^{M-1}\bar{u}^M(x) \geq 0, \quad \text{on} \quad G^{M-1} , \tag{2.34}$$

$$\text{(c)} \qquad [W^{M-1}(x) + I_M^{M-1}\bar{u}^M(x)][L^{M-1}W^{M-1}(x) - S_M^{M-1}\bar{r}^M(x)] = 0, \quad \text{on} \quad G^{M-1} ,$$

$$\text{(d)} \qquad W^{M-1}(x) = 0, \quad \text{on} \quad \partial G^{M-1} .$$

Here I_M^{M-1} and S_M^{M-1} are operators taking grid functions on G^M into grid functions on G^{M-1}. (As an aid in memorization, note that in $I_M^{M-1}\bar{u}^M$ the subscript M and superscript M "cancel".)

The operators I_M^{M-1} and S_M^{M-1} can be defined in many ways. One choice is to choose both I_M^{M-1} and S_{M-1}^M to be the injection operator:

$$\mathrm{Inj}_M^{M-1} w(x) = w(x), \quad x \in G^{M-1} \tag{2.35}$$

Other choices for I_M^{M-1} and S_M^{M-1} will be discussed later.

If we were solving a linear boundary value problem then condition (2.34b) would not apply and it would be most efficient to solve for the correction W^{M-1} on G^{M-1}. Since we are solving inequalities the problem is nonlinear and it is necessary to solve for a 'full approximation' $\bar{U}^{M-1}$ on G^{M-1}.

Setting

$$\bar{U}^{M-1}(x) = W^{M-1}(x) + I_M^{M-1}\bar{u}^M(x) , \tag{2.36}$$

it follows that $\bar{U}^{M-1}(x)$ satisfies the LCP

$$\begin{aligned}
&\text{(a)} && L^{M-1}\bar{U}^{M-1}(x) \leq \bar{F}^{M-1}(x), && \text{in } G^{M-1} ,\\
&\text{(b)} && \bar{U}^{M-1}(x) \geq 0, && \text{in } G^{M-1} ,\\
&\text{(c)} && \bar{U}^{M-1}(x)[L^{M-1}\bar{U}^{M-1}(x) - \bar{F}^{M-1}(x)] = 0, && \text{in } G^{M-1} ,\\
&\text{(d)} && \bar{U}^{M-1}(x) = g(x), && \text{on } \partial G^{M-1} ,
\end{aligned} \tag{2.37}$$

where

$$\bar{F}^{M-1}(x) = S_M^{M-1}\bar{r}^M(x) + L^{M-1}I_M^{M-1}\bar{u}^M(x) = S_M^{M-1}[\bar{F}^M(x) - L^M\bar{u}^M(x)] + L^{M-1}I_M^{M-1}\bar{u}^M(x) . \tag{2.38}$$

Finally, we set

$$\epsilon^{M-1} = \delta \| \nabla \bar{u}^M \|_G , \tag{2.39}$$

and

$$\bar{u}^{M-1} = I_M^{M-1}\bar{u}^M , \tag{2.40}$$

where δ is a constant; in our computations δ has been set equal to .15.

To recapitulate, starting with initial values of $\bar{u}^M$, ϵ^M, and $\bar{F}^M$, we first carry out G^M projected sweeps until convergence slows down. We then introduce a subsidiary problem on G^{M-1} with known $\bar{F}^{M-1}$ and ϵ^{M-1} and initial approximation $\bar{u}^{M-1}$. The process can be repeated, so that at any one stage of the computation we have a sequence of grid approximations $\bar{u}^M, \bar{u}^{M-1}, \ldots, \bar{u}^{k-1}$, (approximating $\bar{U}^M, \bar{U}^{M-1}, \ldots, \bar{U}^{k-1}$, respectively), tolerances $\epsilon^M, \epsilon^{M-1}, \ldots, \epsilon^{k-1}$, and right hand sides $\bar{F}^M, \bar{F}^{M-1}, \ldots, \bar{F}^{k-1}$.

In the general case, $\bar{U}^k$ is the solution of the LCP

$$\begin{aligned}
&\text{(a)} && L^k\bar{U}^k(x) \leq \bar{F}^k(x), && \text{in} \quad G^k, \\
&\text{(b)} && \bar{U}^k(x) \geq 0, && \text{in} \quad G^k, \\
&\text{(c)} && \bar{U}^k(x)(L^k\bar{U}^k(x) - \bar{F}^k(x)) = 0, && \text{in} \quad G^k, \\
&\text{(d)} && \bar{U}^k(x) = g(x) && \text{on} \quad \partial G^k;
\end{aligned} \tag{2.41}$$

or equivalently,

$$\begin{aligned}
&\text{(a)} && A^k\bar{U}^k \leq \bar{b}^k, \\
&\text{(b)} && \bar{U}^k \geq 0, \\
&\text{(c)} && (\bar{U}^k)^T(A^k\bar{U}^k - b^k) = 0.
\end{aligned} \tag{2.42}$$

This LCP is solved approximately using G^k projected sweeps until the latest approximation $\bar{u}^k$ satisfies either

$$\|\bar{v}\bar{u}^k\|_G \leq \epsilon^k, \tag{2.43}$$

or

$$\|\bar{v}\bar{u}^k\|_G \geq \eta\|\bar{v}\bar{u}^k_{old}\|_G. \tag{2.44}$$

If (2.44) holds but (2.43) does not then a new problem on G^{k-1} is defined by setting

$$\bar{F}^{k-1} = S_k^{k-1}[\bar{F}^k - L^k\bar{u}^k] + L^{k-1}I_k^{k-1}\bar{u}^k, \tag{2.45}$$

$$\epsilon^{k-1} = \delta\|\bar{v}\bar{u}^k\|_G, \tag{2.46}$$

$$\bar{u}^{k-1} = I_k^{k-1}\bar{u}^k, \tag{2.47}$$

$$\bar{U}^{k-1} = W^{k-1} + I_k^{k-1}\bar{u}^k, \tag{2.48}$$

$$V^k = \bar{U}^k - \bar{u}^k, \tag{2.49}$$

where W^{k-1} is an approximation to V^k on G^{k-1}. Unless otherwise indicated, I_k^{k-1} and S_k^{k-1} will be taken to be the injection operator Inj_k^{k-1}.

At some stage the latest approximation $\bar{u}^{k-1}$ must satisfy, (2.43)

$$\|\bar{u}^{k-1}\|_G \leq \epsilon^{k-1}, \tag{2.50}$$

if for no other reason than that when $k-1=1$ we cannot introduce any more sub sidiary problems and must iterate until (2.50) is satisfied. Having found an approximation $\bar{u}^{k-1}$ of sufficient accuracy, we return to G^k. To do so, we first determine an approximation w^{k-1} to W^{k-1} from (2.48) namely

$$w^{k-1} = \bar{u}^{k-1} - I_k^{k-1}\bar{u}^k . \tag{2.51}$$

Next, let I_{k-1}^k be an interpolation operator taking grid functions on G^{k-1} into grid functions on G^k. A possible choice for I_{k-1}^k is the linear interpolation operator L_{k-1}^k defined as follows. If P_1, P_2, P_3, and P_4 are the corners of a square in G^{k-1} (see Figure 2.1) then

$$L_{k-1}^k w^{k-1}(P_i) = \begin{cases} w^{k-1}(P_i), & 1 \le i \le 4 , \\ (w^{k-1}(P_1) + w^{k-1}(P_2))/2, & i = 5 , \\ (w^{k-1}(P_1) + w^{k-1}(P_4))/2, & i = 6 , \\ \left(\sum_{i=1}^{4} w^{k-1}(P_i)\right)/4, & i = 7 . \end{cases} \tag{2.52}$$

(Other choices for I_{k-1}^k will be discussed later.)

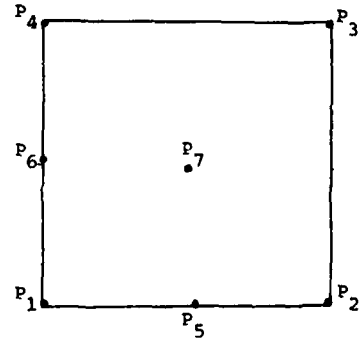


Figure 2.1: Linear interpolation from G^{k-1} to G^k.

Since W^{k-1} is an approximation to V^k on G^{k-1},

$$I_{k-1}^k w^{k-1} = I_{k-1}^k [\bar{u}^{k-1} - I_k^{k-1}\bar{u}^k] , \tag{2.53}$$

is an approximation to v^k, and, noting (2.49),

$$\tilde{u}^k = \bar{u}^k + I_{k-1}^k w^{k-1} , \tag{2.54}$$

is an improved approximation to $\bar{u}^k$. However, because of the nonnegativity constraint upon $\bar{u}^k$, we allow somewhat greater generality and replace $\bar{u}^k$ as follows:

$$\bar{u}^k \leftarrow \varphi(\tilde{u}^k;\bar{u}^k) = \varphi(\bar{u}^k + I_{k-1}^k w^{k-1};\bar{u}^k) . \tag{2.55}$$

Initially we set

$$\varphi(\tilde{u}^k;\bar{u}^k) = \tilde{u}^k , \tag{2.56}$$

but other choices will be considered later.

PFAS is described by (2.24) through (2.56). A flowchart is given in Figure 3.1, and the implementation is discussed in Section 3. If the algorithm converges, we will eventually obtain an approximation $\bar{u}^M$ satisfying the required accuracy condition (2.28) and the algorithm will terminate.

3. IMPLEMENTATION OF PFAS.

The flowchart for PFAS is given in Figure 3.1. PFAS has been implemented as a FORTRAN subroutine for the case when Ω is a rectangle in R^2, $\mathcal{L}$ is the Laplacian operator, I_k^{k-1} and S_k^{k-1} are injections (equation (2.35)), and I_{k-1}^k is linear interpolation (equation (2.52)). The subroutine PFAS, which is listed in Appendix A as part of the program for solving the porous flow free boundary problem described in Section 4, is a straightforward modification of an earlier program, FAS Cycle C, of Brandt. In the subroutine PFAS most of the computations are performed by auxiliary subroutines, and the flowchart shows the role played by these auxiliary subroutines.

One reason for giving a listing of PFAS is so that the reader can appreciate how easy it is to implement PFAS. It may also be remarked that many other interesting free boundary problems (for example, elastic-plastic torsion problems and cavitating journal bearing problems) are formulated on simple polygonal regions, and the program given here could easily be modified to handle these problems.

The following comments arise:

1. In PFAS, the LCP for $\bar{U}^k$ is solved in the form (2.42) rather than (2.41), but the values of $\bar{u}^k$ on ∂G^k are also stored. Thus, $\bar{b}^k = h_k^2 \bar{F}^k$ is stored instead of $\bar{F}^k$. In going from G^k to G^{k-1} we have, from (2.45), since $h_{k-1} = 2h_k$,

$$\begin{aligned} \bar{b}^{k-1} &= h_{k-1}^2 \bar{F}^{k-1}, \\ &= h_{k-1}^2 (S_k^{k-1}[\bar{F}^k - L^k \bar{u}^k] + L^{k-1} I_k^{k-1} \bar{u}^k), \\ &= h_{k-1}^2 (S_k^{k-1} h_k^{-2}[\bar{b}^k - A^k \bar{u}^k] + L^{k-1} I_k^{k-1} \bar{u}^k), \\ &= 4 S_k^{k-1}[\bar{b}^k - A^k \bar{u}^k] + A^{k-1} I_k^{k-1} \bar{u}^k. \end{aligned} \tag{3.1}$$

2. A G^k-work-unit is the work required for one G^k projected sweep. The work for one G^k projected sweep is approximately $2^{-n(M-k)}$ G^M-work-units, and PFAS keeps track of the total number of G^M-work-units, WU. When no confusion is possible we write "work unit" instead of "G^M-work-unit".

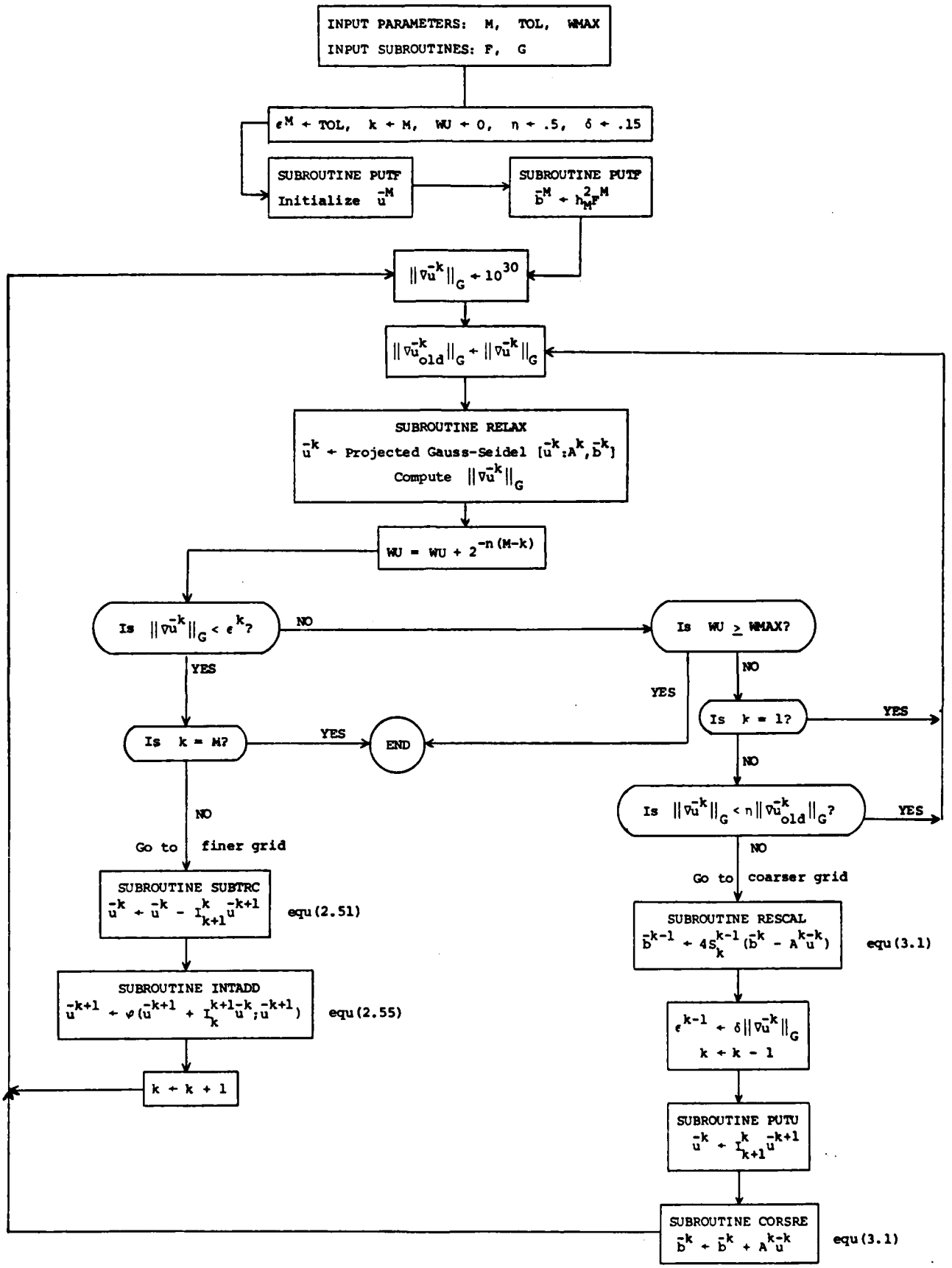


Figure 3.1: Flow Chart for PFAS.

3. The asymptotic speed of convergence is measured by the asymptotic convergence factor $\mathring{\mu}$, which is defined by

$$\mathring{\mu} = \lim_{WU \to \infty} \left[\| \nabla \bar{u}^{-M} \|_G \right]^{1/WU} . \tag{3.2}$$

4. All the numerical computations were performed on the Univac 1180 at the University of Wisconsin-Madison. The programs were written in ASCII FORTRAN and compiled and executed using full optimization.

The Univac 1180 single-precision arithmetic has approximately eight decimals. The residuals usually decrease quite rapidly at the beginning of a computation so the round-off threshhold is quickly reached. For example, for the problem considered in Section 4 with $M = 5$, $\| u^M \|_G$ is about 2×10^3 and the single precision algorithm went into a loop when $\| \nabla \bar{u}^{-M} \|_G$ reached 5×10^{-6} after a mere 50 work units.

In the numerical experiments we were particularly interested in measuring the asymptotic convergence factor $\mathring{\mu}$. To eliminate round-off effects, all the computations reported on here used double precision arithmetic. Of course, this is not normally necessary. Furthermore, even if very accurate solutions of the discrete problem (2.2) were required, it would suffice to store $\bar{u}^{-M}$ in double precision and all other quantities in single precision.

The execution times quoted are those provided by the Univac 1180 Exec. System. As is often the case on timesharing systems, the times are only reproducible to within about 10%.

Because of its word length, the UNIVAC 1180 can only directly access 64K words of storage. When $M \geq 7$ more than 64K words of storage are needed by PFAS and there is a significant degradation in performance.

5. To measure $\mathring{\mu}$ the iterations were continued for the first 100 work units, unless the residuals vanished before. In practice one usually iterates only for about 30 work units.

We also used several values of M in order to measure the dependence of $\mathring{\mu}$ upon M.

Part of the output of a typical computation using PFAS is shown in Figure 3.2. After each G^k projected sweep, the values of the level k, the residual norm $\|\bar{\nabla u}^k\|_G$, and the number of work units WU are printed out.

The computations starting at a level M/level (M - 1) junction and continuing until the next level M/level (M - 1) junction are called a cycle (see Figure 3.2). For the cycle shown in Figure 3.2, $\|\bar{\nabla u}^M\|_G$ decreased from $.293\ 10^{-9}$ to $.110\ 10^{-9}$ with the expenditure of (99.039-94.400) = 4.639 work units.

While minor variations do arise, a cycle often consists of a sequence of 2 sweeps at each of levels $M - 1, M - 2, \ldots, 1$, followed by 2 sweeps at each of levels $2, \ldots, M - 1$, terminating with 2 or 3 sweeps at level M. If this pattern is followed with 3 sweeps at level M then the average number of work units per cycle is

$$3 + 4[2^{-n} + 2^{-2n} + \ldots] = 3 + 4/(2^n - 1) , \tag{3.3}$$

and the average number of work units per G^M projected sweep is $1 + 4/(3(2^n - 1))$.

Of course, very irregular patterns are observed when the round-off threshhold is reached.

6. As can be seen from Figure 3.2, $\|\bar{\nabla u}^M\|_G$ decreases steadily but not very regularly, in part because of slight variations in the number of sweeps at each level. To evaluate the algorithm, we have used two quantities:

$$r_f = \|\bar{\nabla u}^M_{final}\|_G = \text{the value of } \|\bar{\nabla u}^M\|_G \text{ at the end of the last complete cycle before 100 work units,} \tag{3.4}$$

$$\mathring{\mu}_f = \left[\|\bar{\nabla u}^M_{final}\|_G / \|\bar{\nabla u}^M_{initial}\|_G\right]^{1/[WU_{final} - WU_{initial}]} , \tag{3.5}$$

where $\|\bar{\nabla u}^M_{initial}\|_G$ is the value of $\|\bar{\nabla u}^M\|_G$ after the first G^M sweep. $\mathring{\mu}_f$ is an estimate for the asymptotic convergence factor $\mathring{\mu}$.

For example, for the data in Figure 3.2, the value of $\|\bar{\nabla u}^M_{initial}\|_G$ (which is not shown in Figure 3.2) was 4.95 and, of course, $WU_{initial} = 1$. Thus,

```
LEVEL 5   RESIDUAL NORM=  .755-010   WORK= 91.400
LEVEL 6   RESIDUAL NORM=  .126-008   WORK= 92.400
LEVEL 6   RESIDUAL NORM=  .515-009   WORK= 93.400
LEVEL 6   RESIDUAL NORM=  .293-009   WORK= 94.400
**********END OF CYCLE**********MU =     .7771
LEVEL 5   RESIDUAL NORM=  .196-009   WORK= 94.650
LEVEL 5   RESIDUAL NORM=  .133-009   WORK= 94.900
LEVEL 4   RESIDUAL NORM=  .879-010   WORK= 94.963
LEVEL 4   RESIDUAL NORM=  .613-010   WORK= 95.025
LEVEL 3   RESIDUAL NORM=  .385-010   WORK= 95.041
LEVEL 3   RESIDUAL NORM=  .257-010   WORK= 95.057
LEVEL 2   RESIDUAL NORM=  .133-010   WORK= 95.061
LEVEL 2   RESIDUAL NORM=  .717-011   WORK= 95.064
LEVEL 1   RESIDUAL NORM=  .243-011   WORK= 95.065
LEVEL 1   RESIDUAL NORM=  .447-012   WORK= 95.066
LEVEL 2   RESIDUAL NORM=  .303-011   WORK= 95.070
LEVEL 3   RESIDUAL NORM=  .189-010   WORK= 95.086
LEVEL 3   RESIDUAL NORM=  .714-011   WORK= 95.102
LEVEL 4   RESIDUAL NORM=  .686-010   WORK= 95.164
LEVEL 4   RESIDUAL NORM=  .255-010   WORK= 95.227
LEVEL 4   RESIDUAL NORM=  .138-010   WORK= 95.289
LEVEL 5   RESIDUAL NORM=  .151-009   WORK= 95.539
LEVEL 5   RESIDUAL NORM=  .534-010   WORK= 95.789
LEVEL 5   RESIDUAL NORM=  .284-010   WORK= 96.039
LEVEL 6   RESIDUAL NORM=  .473-009   WORK= 97.039
LEVEL 6   RESIDUAL NORM=  .194-009   WORK= 98.039
LEVEL 6   RESIDUAL NORM=  .110-009   WORK= 99.039
**********END OF CYCLE**********MU =     .7787
LEVEL 5   RESIDUAL NORM=  .737-010   WORK= 99.289
LEVEL 5   RESIDUAL NORM=  .499-010   WORK= 99.539
LEVEL 4   RESIDUAL NORM=  .331-010   WORK= 99.602
LEVEL 4   RESIDUAL NORM=  .231-010   WORK= 99.664
LEVEL 3   RESIDUAL NORM=  .145-010   WORK= 99.680
```

Figure 3.2: Typical Output for the PFAS Algorithm.
(M = 6, Problem (4.1)-(4.2), Run #X67368)

$$r_f \doteq .110\ 10^{-9} ,$$

and

$$\mathring{\mu}_f = \left[\frac{.110\ 10^{-9}}{4.95}\right]^{1/(99.039-1)} \doteq .7787 .$$

We usually only quote r_f to one decimal place and $\mathring{\mu}_f$ to two decimal places, since this is quite adequate for our purposes.

PFAS computes and prints

$$\mathring{\mu} = \left[\|\nabla \bar{u}^M\|_G / \|\nabla \bar{u}^M_{\text{initial}}\|_G\right]^{1/[WU-WU_{\text{initial}}]} \tag{3.6}$$

at the end of each cycle.

7. In all the experiments reported here the parameters δ and η (see (2.29) and (2.39)) were given by $\delta = .5$ and $\eta = .15$. According to Brandt [77] the rate of convergence is not very sensitive to changes in these parameters, and this was confirmed in a few experiments.

In a few cases, but never for $\delta = .5$ and $\eta = .15$, the program "hunted": that is, the program went down from G^M to G^1, up to G^k for $k < M$, and then down again to G^1 instead of continuing up to G^M. This might happen several times before G^M was reached again.

4. NUMERICAL RESULTS FOR POROUS FLOW THROUGH A DAM.

Calculations were performed on the well-known free boundary problem describing the flow of water through a porous dam. The geometry is shown in Figure 4.1. Water seeps from a reservoir of height y_1 through a rectangular dam of width a to a reservoir of height y_2. Part of the dam is saturated and the remainder of the dam is dry. The wet and dry regions are separated by an unknown free boundary which must be found as part of the solution. For an introduction to the problem see Bear [1972], or Cryer [1976].

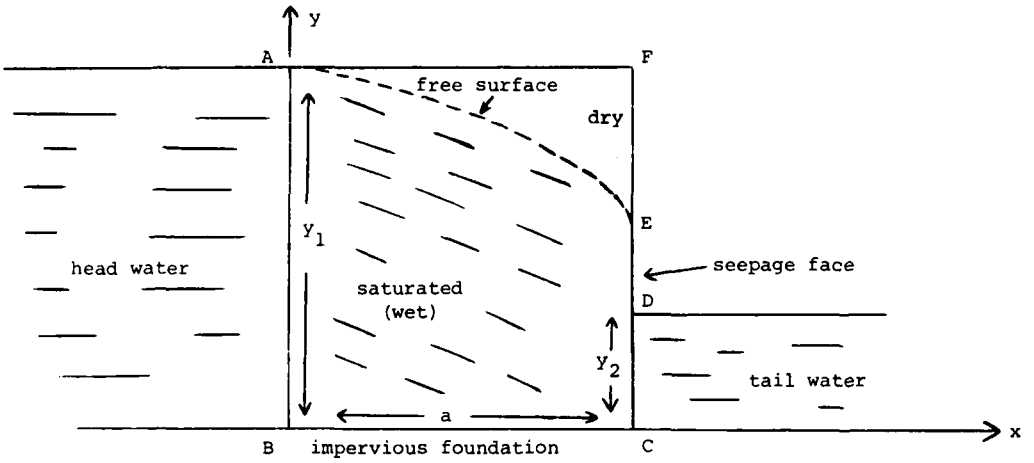


Figure 4.1 Seepage Through a Simple Rectangular Dam

As shown by Baiocchi [1971] the problem can be formulated as follows: Find u on the rectangle $\Omega = ABCF$ such that

$$\begin{aligned} \nabla^2 u &\leq 1, \quad \text{on } \Omega, \\ u &\geq 0, \quad \text{on } \Omega, \\ u(\nabla^2 u - 1) &= 0, \quad \text{on } \Omega, \end{aligned} \tag{4.1}$$

$$u = g = \begin{cases} (y_1 - y)^2/2 & \text{on AB}, \\ (y_2 - y)^2/2 & \text{on CD}, \\ [y_1^2(a - x) + y_2^2(x)]/2a, & \text{on BC}, \\ 0, & \text{on DFA}, \end{cases} \tag{4.2}$$

which is in the form (1.1).

This problem was solved using PFAS, with I_k^{k-1} and S_k^{k-1} being injections (equation (2.35)) and with I_{k-1}^k defined by linear interpolation (equation (2.52)). The initial values of $\tilde{u}^M$ were obtained by interpolating the boundary values of u linearly in the x direction. A listing of the program is given in Appendix A.

We considered the well-known case, $y_1 = 24$, $y_2 = 4$, and $a = 16$. In all computations G^1 was a $(2 + 1) \times (3 + 1)$ grid with $h_1 = 8$. The finest grid used was G^7 with $(128 + 1) \times (192 + 1) = 24897$ grid points.

To give the reader an idea of the solution, the solution U^2 of (2.2) is given to four decimal places in Table 4.1.

y \ x	0	4	8	12	16
24	0	0	0	0	0
20	8	2.5371	0	0	0
16	32	18.1486	6.7841	0	0
12	72	47.2732	24.9879	7.9120	0
8	128	89.9564	53.9823	22.6601	0
4	200	146.5702	94.3247	44.7462	0
0	288	218.0000	148.0000	78.0000	8

Table 4.1. U^2 for the Dam Problem
(Run #X34654)

The numerical results, for different values of M, and $\epsilon^M = \text{TOL} = 0$, are given in Table 4.2. The most important conclusions are that convergence always occurred and that the convergence factor $\mathring{\mu}_f$ is always less than .81.

Run #	X34654	X34654	X34654	X34654	X34654	PC 3567
M	2	3	4	5	6	7
G^M	5 × 7	9 × 13	17 × 25	33 × 49	65 × 97	129 × 193
r_f	0*	4(-17)*	1(-13)	1(-8)	1(-10)	1(-7)
$\mathring{\mu}_f$	.404	.607	.726	.813	.778	.81
Execution Time for 100 Work Units (Seconds)	.114	.428	1.04	3.55	13.39	**
ρ_{SORopt}	.18	.49	.71	.84	.92	.96

Table 4.2. Solution of the dam problem using PFAS

*Reached round-off level before 100 work units.

**Required 70K workspace so extended storage facility invoked, and timing not compatible.

We now compare the convergence factors $\mathring{\mu}_f$ in Table 4.2 with those for other methods of solving the LCP (2.2).

A popular method of solving the LCP (1.3) is G^M projected SOR (point SOR with projection) which has also been called "modified SOR" by Cottle.

When using G^M projected SOR it is observed experimentally that the values of $\bar{u}^M$ settle down quite quickly into positive values and zero values. Thereafter G^M projected SOR is equivalent to using point SOR on the subset $G^M_+ = \{x \in G^M : U^M(x) > 0\}$. Thus the asymptotic convergence factor for G^M projected SOR is in general equal to the asymptotic convergence factor for point SOR on G^M_+. It is known (Varga [1962, p. 294]) that for a region of area A and for the finite difference equations corresponding to the five-point difference approximation to Laplace's equation with stepsize h, the convergence factor for the optimum choice of overrelaxation parameter ω is approximated quite well by

$$\rho_1(h) = \frac{2}{1 + 3.015[h^2/A]^{\frac{1}{2}}} - 1 . \tag{4.3}$$

In the present case we do not know the area of G_+^M, but, as a rough guide, the area of G_+^M is approximately equal to the area of Ω, which is about 80% of the area of the rectangle ABCF. Therefore, for our present purposes the asymptotic convergence factor for G^M projected SOR with optimum choice of ω may be taken to be

$$\rho_{SORopt} \doteq \frac{2}{1 + 3.015[h^2/(.8 \times 16 \times 24)]^{\frac{1}{2}}} - 1 \doteq \frac{2}{1 + .172\ h} - 1 , \qquad (4.4)$$

and these values are given at the bottom of Table 4.2.

As Table 4.2 shows, for large problems, PFAS is faster than G^M projected SOR. On G^7, for example, the increase in speed (measured in work units) is $\ell n.96/\ell n.81 \doteq 5.2$. Against this, two factors must be borne in mind: (1) PFAS is more complicated and requires more overhead per work unit; (2) PFAS requires somewhat more storage. We discuss these two factors below, but before doing so we wish to emphasize that although these factors reduce the advantage in speed of PFAS, the measured execution times for PFAS are much smaller than those for G^M projected SOR (see Tables 5.3 and 6.3).

1. Overhead.

To obtain an indication of the additional overhead required by PFAS, we compared execution times for $M = 5$. We first used PFAS with $\epsilon^M = 2.10^{-8}$. This required 96.156 work units and took 3.40 seconds. We then modified PFAS so that only the grid $k = M$ was used and so that over-relaxation was used with the over-relaxation parameter ω given by equation (4.4). We were thus using G^M projected SOR with a nearly optimum ω. To reduce $\|\bar{v}u^M\|_G$ to $\epsilon^M = 2.10^{-8}$ required 146 work units and took 4.82 seconds. Since

$$(3.40/96.156)/(4.82/146) \doteq 1.07$$

we conclude that, in this application, the additional overhead required by PFAS only increases the computation time per G^M work unit by about 10%.

2. Storage.

As implemented here, PFAS keeps the solutions and residuals on all the grids, and therefore requires storage for $2[1 + 4^{-1} + 4^{-2} + \ldots] = 8/3$ G^M grids. In contrast, G^M projected SOR requires storage for only one G^M grid.

If storage is at a premium, the residuals on G^M need not be stored and PFAS requires only 5/3 times as much storage as G^M projected SOR. If $\bar{u}^M$ is stored to double precision, but $\bar{u}^k$ and $\bar{b}^k$ are stored to single precision for $k < M$, only 4/3 times as much storage is needed. If $F(x)$ were not the constant 1, but a complicated function, then either the function values or the residuals would have to be stored for G^M projected SOR, and PFAS would require at most 33% more storage. Finally, the PFMG algorithm described in Section 6 often need not store any data on the G^M grid (see Section 6).

Another possible algorithm for solving the LCP (1.3) is the MBSOR (modified block SOR) algorithm of Cottle and Sacher [1978]. This algorithm is based upon the solution of a sequence of "one-dimensional" LCP's in much the same way that line SOR is based upon solving a sequence of "one-dimensional" equations. We used MBSOR to solve the dam problem (4.1), (4.2), for the case $M = 5$. The program was kindly provided by Professor Sacher. We tried a few values of the over-relaxation parameter ω, and found that 1.8 gave the best results. With $\omega = 1.8$ MBSOR required 114 iterations to reduce $\|\nabla u^M\|_G$ to below 2.10^{-8} and took 13.13 seconds. The following comments arise:

1. In numerical experiments on the dam problem, Cottle [1974] found that MBSOR was about 20% faster than "modified point SOR", that is, G^M projected SOR. This is consistent with the fact that, for equations, the convergence ratio for line SOR is only faster by a factor of $\sqrt{2}$ than point SOR while there is more computation per iteration. This is also consistent with the present results, since G^M projected SOR required 146 iterations to reduce the residual to 2.10^{-8} while MBSOR required only 114.

2. The poor execution time of MBSOR (13.13 seconds) compared to PFAS (3.40 seconds) can be explained in part by two factors: (a) MBSOR requires more computation per iteration than is needed by PFAS for a single work unit; (b) the MBSOR program was

written for the case of general coefficients, while the PFAS program takes advantage of the properties of the five-point difference operator.

3. It must also be borne in mind that Cottle and Sacher [1978] found that MBSOR was three times as fast as G^M projected SOR for the journal bearing problem where the solution is zero at a high percentage of the gridpoints.

We conclude from Table 4.2 and from the above discussion, that for the dam problem (4.1), (4.2) *PFAS is faster than* G^M projected SOR and modified block SOR for $M \geq 5$, that is, for grids of dimension 33×49 or greater. Furthermore, we also conclude that the values of $\mathring{\mu}_f$ and ρ_{SORopt} in Table 4.2 provide a reasonably accurate guide to the relative performance of PFAS and G^M projected SOR. We believe that PFAS will be faster than both G^M projected SOR and MBSOR for a wide range of problems.

4. For a grid G^M with N gridpoints, both G^M-projected SOR and modified block SOR have computation times which are $O(N^{3/2})$. As Table 4.2 shows, the computation time for PFAS is $O(N)$. Therefore, the performance of PFAS vis-a-vis the other methods improves as the grids become finer.

5. ALTERNATIVE IMPLEMENTATIONS OF PFAS.

In this section we discuss alternative implementations of PFAS, the best of which achieves substantially improved performance.

The improvement in PFAS which might be possible is suggested by considering the asymptotic convergence ratio, $\mathring{\mu}_{FAS}$ say, for FAS for Poisson's equation. For FAS, the error reduction per G^M-sweep is .5. If each G^M-sweep is accompanied by, on average, one G^k sweep for $1 \leq k \leq M - 1$, then the number of work units per G^M-sweep is

$$1 + 2^{-2} + 2^{-4} + \ldots = 4/3$$

and the convergence ratio is $(.5)^{3/4} = .595$, as stated by Brandt [1977, p. 351]. In the present case, as observed in Section 3, the average number of work units per G^M sweep is

$$1 + 4/[3(2^n - 1)] = 13/9 ,$$

so that

$$\mathring{\mu}_{FAS} = (.5)^{9/13} = .6188 . \tag{5.1}$$

This value of $\mathring{\mu}_{FAS}$ is observed experimentally. The worst observed value of $\mathring{\mu}_f$ for the PFAS results quoted in Section 3 was $\mathring{\mu}_f = .81$. Thus, FAS (for equations) is faster than PFAS (for LCP's) by a factor of $\ln .81/\ln .6188 = 2.28$.

Plausible reasons why PFAS is slower than FAS include the following:

D1: Negative components of $\bar{u}^k$.

The inequality (2.41b) requires that $\bar{u}^k$ be non-negative. In each G^k projected sweep the step (2.7) ensures that $\bar{u}^k$ is non-negative. Furthermore, if I_k^{k-1} is the injection operator the initial approximation $\bar{u}^{k-1}$ defined by (2.47) is also non-negative. However, (2.54) does not preserve non-negativity: in returning to G^k from G^{k-1} the initial approximation $\bar{u}^k$ may have negative components, and this is often observed. Of course, any negative components are removed in the first subsequent G^k projected sweep, but nevertheless the introduction of negative components must retard convergence. □

D2: Large residuals near the free boundary.

At a point $x \in G^k$ where $\bar{U}^k(x) = 0$ the corresponding residual

$$\bar{R}^k(x) = \bar{F}^k(x) - L^k\bar{U}^k(x) \tag{5.2}$$

must be non-negative because of the inequality (2.41a) but need not be small. □

D3: Influence of the discrete interface.

The discrete interface $\Gamma^k \subset R^2$ is the interface between the set of points where $\bar{U}^k > 0$ and the set of points where $\bar{U}^k = 0$. Γ^k approximates the continuous interface, or free boundary, Γ separating the points where the solution $u(x)$ is positive from the points where $u(x)$ is zero.

In special cases it may happen that $\Gamma^k = \Gamma$ for all k, in which case PFAS converges as fast as FAS. An example is given by problem (5.3), (5.4) below with $R = 2$, for which Γ is the line $y = 5 - 2x$; it is found experimentally that $\Gamma^k = \Gamma$ for $k \leq 6$.

In general, Γ^k and Γ differ by $0(h_k)$, and Γ^k and Γ^{k-1} differ by $0(h_k)$. In particular, it may happen that $\bar{U}^k(x) > 0$ while $\bar{U}^{k-1}(x) = 0$. Furthermore, near Γ^{k-1} the residuals may be less smooth because of the projection (2.7) and because of the irregular shape of Γ^k and Γ^{k-1}. This introduces errors in the coarse grid corrections (2.55) thereby slowing the rate of convergence. Finally, the injection operator (2.35) is not adequate if the data to which it is applied is not smooth. □

Multigrid algorithms can often be speeded up by modifying the operators I_k^{k-1}, S_k^{k-1}, and I_{k-1}^k. We have tried a number of modifications of the corresponding PFAS subroutines which were intended to address the difficulties D1 to D3 mentioned above.

The subroutine PFAS in Appendix A was modified so as to facilitate experimentation. This was done by changing the calls to the auxiliary subroutines so that input "switch" parameters determined which version of each subroutine was used.

In addition, computations were also made for the following problem:

$$\begin{aligned} &(a) \qquad \nabla^2 u \le f(x,y), \quad \text{in} \quad \Omega , \\ &(b) \qquad u \ge 0, \quad \text{in} \quad \Omega , \\ &(c) \qquad u = g, \quad \text{on} \quad \partial\Omega , \end{aligned} \tag{5.3}$$

where $\Omega = [0,3] \times [0,2]$, and where f and g are chosen so that the exact solution is

$$u = [\cos(x+y) + 2][\max\{0; 2.5\ R - Rx - y\}]^2 . \tag{5.4}$$

Here, R is a parameter which is chosen close to the value 2. Note that $u \in C^2(\Omega)$ and $u = 0$ above the line $y = R(2.5 - x)$. By changing the value of R we can force gridpoints to lie very close to the exact free boundary; this may be expected to cause PFAS difficulty because if $\bar{U}^k(x)$ is positive but very small for some $x \in G^k$ then it will take PFAS a large number of iterations to determine whether $\bar{U}^k(x)$ is zero or positive.

The modified version of PFAS is called PFASMD and is listed in Appendix B as part of a program for solving the porous flow problem of Section 4 and problems (5.3), (5.4). PFASMD was used to compute all the results in this section.

Our first modifications to the auxiliary subroutines of PFAS were not very successful, but they were very instructive and we briefly summarize them. In all cases, the results are for the dam problem with $M = 5$. (All were with run #X35519).

M1. PFAS was modified so as to enforce nonnegativity of $\bar{u}^k$ immediately after returning from G^{k-1}. This was done by defining φ in (2.55) by

$$\varphi(\bar{u}^k; \bar{u}^k) = \max\{0, \bar{u}^k\} . \tag{5.5}$$

The new subroutine was called INTAPR.

This modification converged slightly faster than PFAS with $\mathring{\mu}_f = .803$. □

M2. The usual situation in which the nonnegativity of $\bar{u}^k$ is violated is as follows.

Let $\bar{u}^k(x) = 0$, where $x \in G^k$ but $x \notin G^{k-1}$. Let $y \in G^{k-1}$ be a neighbor of x, such that $\bar{u}^k(y) > 0$. It may then happen that $w^{k-1}(y) < 0$. As a result, $(I^k_{k-1} w^{k-1})(x)$ may be negative, and if so the updated value of $\bar{u}^k(x)$ will be negative.

To avoid this, PFAS was modified by changing the subroutines SUBTRC and PUTU so that the operator I_k^{k-1} became

$$I_k^{k-1}\bar{u}^k(y) = \begin{cases} \bar{u}^k(y) & \text{if } \bar{u}^k(x) > 0 \text{ for all eight} \\ & \text{neighbors } x \text{ of } y \text{ in } G^k, \\ 0, & \text{otherwise .} \end{cases} \tag{5.6}$$

The new subroutines were called PUTUNN and SUBTNN, respectively.

Remembering from (2.48) that

$$\bar{U}^{k-1} = W^{k-1} + I_k^{k-1}\bar{u}^k ,$$

we see from (5.6) and (2.41b) that the restraint $W^{k-1}(y) \geq 0$ is enforced for every point $y \in G^{k-1}$ with a neighbor $x \in G^k$ such that $\bar{u}^k(x) = 0$.

This modification converged slightly more slowly than PFAS, with $\mathring{\mu}_f = .817$. □

M3. PFAS was modified so that if the current value of $\bar{u}^M(x)$ was zero, then $\bar{u}^k(x)$ was forced to be zero for $k < M$. This was done by changing the subroutine RELAX. In effect, (2.7) was followed by a further operation:

$$\text{If } k < M \text{ and } \bar{u}^M(x_j^k) = 0 \text{ then } \bar{u}_j^{k,s} = 0 . \tag{5.7}$$

The new subroutine was called RELXFR.

This modification converged but much more slowly than PFAS with $\mathring{\mu}_f = .887$ □

M4. Brandt [1977, p. 378] has found residual weighting useful when the coefficients of the differential equation are changing rapidly. We, therefore, changed the subroutine RESCAL so that S_k^{k-1} became:

$$4\, S_k^{k-1} r^k(x) = \sum_{\Delta} \rho(\Delta) r^k(x + \Delta h_k) , \tag{5.8}$$

where $\Delta = (\Delta_1, \Delta_2)$ for integers Δ_1, Δ_2 and the only nonzero $\rho(\Delta)$ are

$$\begin{aligned} &\rho(0,0) = 1 , \\ &\rho(0,1) = \rho(1,0) = \rho(0,-1) = \rho(-1,0) = \tfrac{1}{2} , \\ &\rho(1,1) = \rho(1,-1) = \rho(-1,1) = \rho(-1,-1) = \tfrac{1}{4} . \end{aligned} \tag{5.9}$$

The new subroutine was called RESCAV.

This modification cycled between G^1 and G^2, as did also the further modification for which I_k^{k-1} was also defined by (5.8), (5.9).

The nonconvergence of the modification M4 requires explanation, and this is provided by

Lemma 5.1.

Let φ be defined by (2.56). For $1 \leq k \leq M$ let $\bar{U}^k$ be the solution of the LCP (2.41), where $\bar{F}^k$ satisfies (2.45). Finally, let I_{k-1}^k satisfy

$$(I_{k-1}^k(z^{k-1}) = 0) \Rightarrow (z^{k-1} = 0), \quad \text{for all } z^{k-1} \in R^{N_{k-1}} . \tag{5.10}$$

Then for PFAS to converge it is necessary that

$$S_k^{k-1}[\bar{F}^k - L^k\bar{U}^k] \geq 0 , \tag{5.11}$$

$$I_k^{k-1}\bar{U}^k \geq 0 , \tag{5.12}$$

$$[I_k^{k-1}\bar{U}^k]^T S_k^{k-1}[\bar{F}^k - L^k\bar{U}^k] = 0 . \tag{5.13}$$

Proof: We apply PFAS by setting $\bar{u}^k = \bar{U}^k$, and forming the LCP (2.41) on G^{k-1}:

$$\begin{aligned} L^{k-1}\bar{U}^{k-1} &\leq \bar{F}^{k-1} , \\ \bar{U}^{k-1} &\geq 0 , \qquad (*) \\ (\bar{U}^{k-1})^T(L^{k-1}\bar{U}^{k-1} - \bar{F}^{k-1}) &= 0 . \end{aligned}$$

Solving this exactly so that $\bar{u}^{k-1} = \bar{U}^{k-1}$, we then return to G^k. Since PFAS converges, the new value of $\bar{u}^k$ given by (2.55) must be equal to $\bar{U}^k$. That is,

$$I_{k-1}^k w^{k-1} = I_{k-1}^k[\bar{U}^{k-1} - I_k^{k-1}\bar{U}^k] = 0 ,$$

which, from (5.10), implies that

$$\bar{U}^{k-1} = I_k^{k-1}\bar{U}^k .$$

Substituting into (*) and noting (2.45) we obtain (5.11) through (5.13).

The following remarks follow from Lemma 5.1.

1. Lemma 5.1 brings out an interesting difference between multigrid methods for equations and for inequalities. For equations, $\bar{F}^k - L^k\bar{U}^k = 0$ and conditions (5.11)-(5.13) are satisfied for any reasonable choice of S_k^{k-1} and I_k^{k-1}, but this is not true for inequalities. □

2. Since $\bar{U}^k$ solves (2.41), inequalities (5.11) and (5.12) will certainly hold if S_k^{k-1} and I_k^{k-1} map nonnegative vectors into nonnegative vectors. In particular, this will be the case if S_k^{k-1} and I_k^{k-1} take linear combinations of values with nonnegative weights. □

3. If S_k^{k-1} and I_k^{k-1} are injections, then (5.13) is implied by (2.41c). □

4. If S_k^{k-1} is defined by (5.8) and (5.9) while I_k^{k-1} is injection then (5.13) does not hold in general. This is because in general there will be points $x,y \in G^k$ such that $x \in G^{k-1}$, $\bar{U}^k(x) > 0$, $\bar{U}^k(y) = 0$, y is a neighbor of x in G^k and $(\bar{F}^k - L^k\bar{U}^k)(y) > 0$. Then

$$I_k^{k-1}\bar{U}^k(x) = \bar{U}^k(x) > 0 ,$$

and

$$(S_k^{k-1}(\bar{F}^k - L^k\bar{U}^k))(x) \geq \frac{1}{4}(\bar{F}^k - L^k\bar{U}^k)(y) > 0 ,$$

so that (5.13) does not hold. This explains why the modification M4 of PFAS did not converge. □

We now describe two further modifications of PFAS which were tried:

M5. Bearing Lemma 5.1 in mind it is possible to introduce weighted sums for which (5.13) does hold. One choice uses weighted residuals only near the boundary:

$$4S_k^{k-1}r^k(x) = \begin{cases} 4r^k(x), & \text{if } \bar{u}^k(x) = 0 \text{ or if } \bar{u}^k(y) > 0 \\ & \text{for all eight neighbors } y \in G^k \text{ of } x , \\ \sum_{\Delta} \rho(\Delta) r^k(x + \Delta h_k) \ \text{signum}\,[\bar{u}^k(x + \Delta h_k)] , & \\ \text{otherwise} & \end{cases} \tag{5.14}$$

where

$$\text{signum } \alpha = \begin{cases} 1, & \text{if } \alpha > 0 , \\ 0, & \text{if } \alpha = 0 , \end{cases}$$

and where the weights $\rho(\Delta)$ are as in (5.9). This was done by an appropriate change in the subroutine RESCAL; the new subroutine was called RESCL1.

M6. As mentioned in D1 and D3 above, if $\bar{u}^k(x) = 0$ then it may happen that $\tilde{u}^k(x) = \bar{u}^k(x) + I_{k-1}^k w^{k-1}(x)$ is not zero. It can be argued that changes of $\bar{u}^k(x)$ from or to zero should only be done on G^k. We, therefore, modified the subroutine INTADD so that in (2.55) φ was defined by

$$\varphi(\tilde{u}^k(x);\bar{u}^k(x)) = \begin{cases} \tilde{u}^k(x), & \text{if } \bar{u}^k(x) > 0 , \\ 0, & \text{otherwise} . \end{cases} \tag{5.15}$$

The new subroutine was called INTADM. □

The modifications M5 and M6 are independent, and we solved (4.1), (4.2) with $M = 5$ and different combinations of M5 and M6. In each case, the computations were terminated when $\|\nabla \bar{u}^M\|_G \leq 2\ 10^{-8}$. The results are summarized in Table 5.1.

Modifications	-	M5	M6	M5 and M6
Work Units	96.15	126.12	42.81	43.76
Execution Time (Seconds)	3.40	4.67	1.63	1.76
$\mathring{u}_f$	.815	.854	.623	.623

Table 5.1: Solution of (4.1), (4.2) with $M = 5$ and $\epsilon^M = 2.10^{-8}$ for modifications 5 and 6.
(Run #X35026)

The performance of PFAS is of course problem dependent. In Table 5.2 we compare modifications 5 and 6 for the problem (5.3), (5.4). As in Table 5.1 we iterated until $\|\nabla u^{(k)}\|_G \leq 2.10^{-8}$ on G^5.

Modifications	-	M5	M6	M5 and M6
Work Units	73.62	74.32	56.96	65.57
Execution Time (Seconds)	3.09	3.24	2.58	3.01
$\mathring{\mu}_f$	.731	.738	.669	.704

Table 5.2: Solution of (5.3), (5.4) with $M = 5$, $R = 32/15$ and $\epsilon^M = 2.10^{-8}$ for modifications 5 and 6.
(Run #X35563)

We conclude from the results given in Tables 5.1 and 5.2 that the use of modification 6 yields substantial improvements.

Finally, in Table 5.3 we extend Table 4.2 by comparing the measured execution times for the projected SOR method and the best modification of PFAS (φ defined by (5.15) and S_k^{k-1} defined by injection) for the dam problem for various values of M. In each case, the iterations were continued until $\|\nabla\bar{u}\|_G \leq 2\ 10^{-8}$.

M / G^M		2 / 5×7	3 / 9×13	4 / 17×25	5 / 33×49	6 / 65×97
G^M Projected SOR	G^M iterations	19	34	69	146	295
	Execution Time (seconds)	.02	.09	.60	4.88	39.37
PFASMD (M6)	G^M work units	23	30.5	38.7	42.8	45.7
	Execution Time (seconds)	.04	.12	.41	1.64	6.57

Table 5.3: Comparison of G^M Projected SOR and PFASMD (modification M6) for the dam problem with $M = 5$ and $\epsilon^M = 2.10^{-8}$.
(Run #'s X35584 and X35564)

As can be seen from Table 5.3, PFASMD is better than projected SOR except for very small grids.

6. PFMG (PROJECTED FULL MULTIGRID ALGORITHM)

In this section we describe PFMG (Projected Full Multigrid Algorithm) which is a modification of the Full Multigrid Algorithm of Brandt. The flowchart for PFMG is given in Figure 6.1. PFMG has been implemented as a FORTRAN subroutine for the case when Ω is a rectangle in R^2, and $\mathcal{L}$ is the Laplacian operator. This subroutine is listed in Appendix C as part of the program for solving the porous flow free boundary problem of Section 4, and the problem (5.3), (5.4).

PFMG differs from PFASMD in the following respects.

I. Instead of beginning on G^M, one begins on a coarser grid G^{LIN} and gradually works up to G^M.

The computations begin on the initial grid G^ℓ, $\ell = LIN$, with an initial approximation $\bar{u}^\ell$. $\bar{u}^\ell$ is computed to the required accuracy using grids G^1 through G^ℓ as in the PFASMD implementation of PFAS, except that, as will be discussed below, the decision to move to a different grid is based on slightly different criteria.

Once $\bar{u}^\ell$ has been found to sufficient accuracy, the initial approximation $\bar{u}^{\ell+1}$ is obtained from

$$\bar{u}^{\ell+1} = J_\ell^{\ell+1}\bar{u}^\ell , \tag{6.1}$$

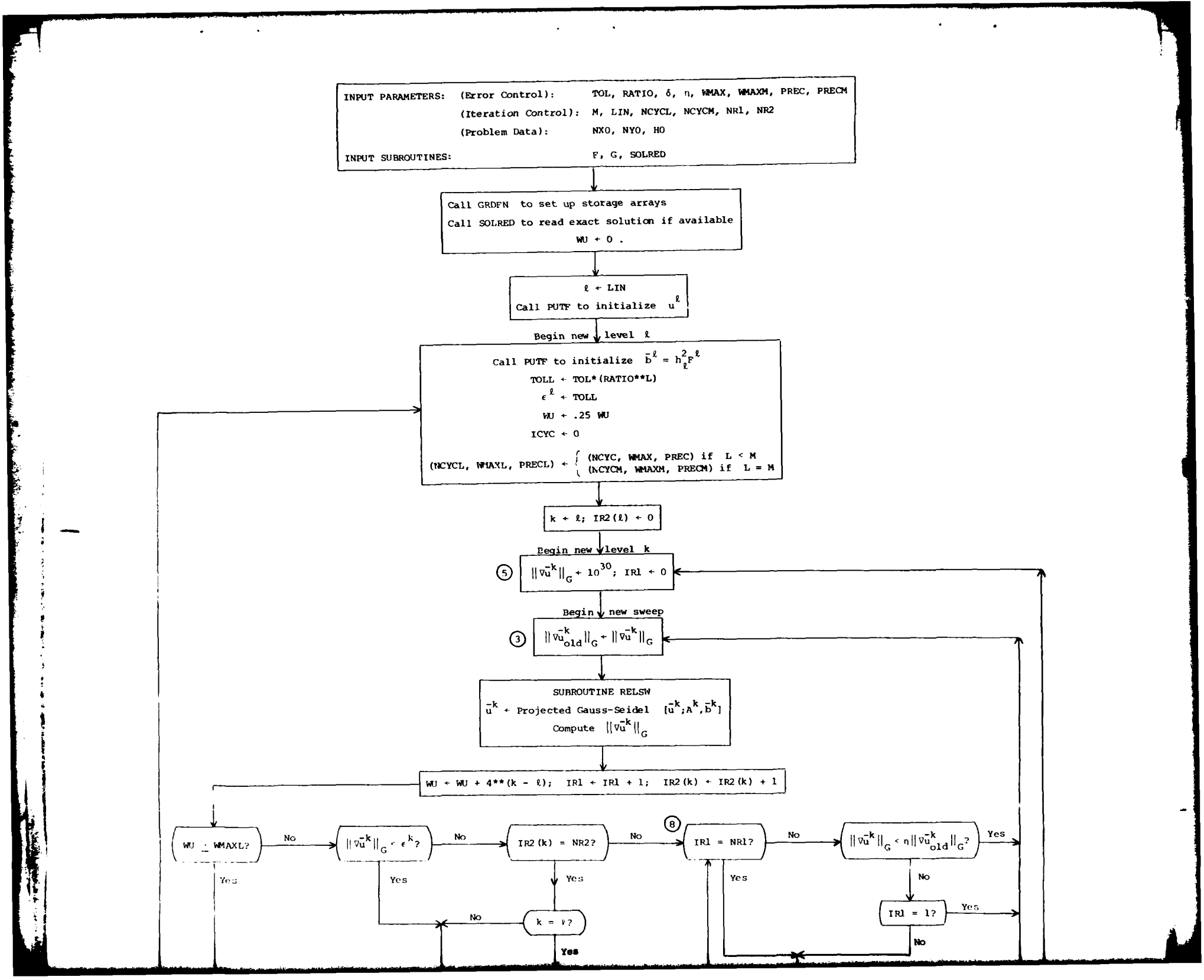
where $J_\ell^{\ell+1}$ is an interpolation operator taking grid functions on G^ℓ into grid functions on $G^{\ell+1}$. It is known (Brandt [1977, p. 377]) that $J_\ell^{\ell+1}$ should be more accurate than $I_\ell^{\ell+1}$ in order to preserve the smoothness of $\bar{u}^\ell$.

In PFMG $J_\ell^{\ell+1}$ is implemented as a subroutine INTRP3 which uses cubic interpolation. (To use INTRP3 we must have $\ell \geq 2$ and so $LIN \geq 2$.) INTRP3 is based upon repeated use of the cubic interpolation formulas

$$f\left(\tfrac{1}{2}\right) = [-f(-1) + 9f(0) + 9f(1) - f(2)]/16 , \tag{6.2}$$

$$f\left(\tfrac{3}{2}\right) = [f(-1) - 5f(0) + 15f(1) + 5f(2)]/16 . \tag{6.3}$$

Repeating this process, we finally obtain an initial approximation $\bar{u}^M$ on G^M. Thereafter, the computation proceeds essentially as in PFASMD.


INPUT PARAMETERS: (Error Control): TOL, RATIO, δ, η, WMAX, WMAXM, PREC, PRECM
(Iteration Control): M, LIN, NCYCL, NCYCM, NR1, NR2
(Problem Data): NX0, NY0, H0
INPUT SUBROUTINES: F, G, SOLRED
Call GRDFN to set up storage arrays
Call SOLRED to read exact solution if available
WU ← 0 .
ℓ ← LIN
Call PUTF to initialize u^{ℓ}
Begin new level ℓ
Call PUTF to initialize $\bar{b}^{\ell} = h_{\ell}^2 F^{\ell}$
TOLL ← TOL*(RATIO**L)
ϵ^{ℓ} ← TOLL
WU ← .25 WU
ICYC ← 0
(NCYCL, WMAXL, PRECL) ← (NCYC, WMAX, PREC) if L < M
(NCYCM, WMAXM, PRECM) if L = M
k ← ℓ; IR2(ℓ) ← 0
Begin new level k
5
$\|\nabla \bar{u}^{k}\|_G \leftarrow 10^{30}$; IR1 ← 0
Begin new sweep
3
$\|\nabla \bar{u}^{k}_{old}\|_G \leftarrow \|\nabla \bar{u}^{k}\|_G$
SUBROUTINE RELSW
$\bar{u}^{k}$ ← Projected Gauss-Seidel $[\bar{u}^{k}; A^{k}, \bar{b}^{k}]$
Compute $\|\nabla \bar{u}^{k}\|_G$
WU ← WU + 4**(k - ℓ); IR1 ← IR1 + 1; IR2(k) ← IR2(k) + 1
WU ≥ WMAXL?
No
$\|\nabla \bar{u}^{k}\|_G < \epsilon^{k}$?
No
IR2(k) = NR2?
No
8
IR1 = NR1?
No
$\|\nabla \bar{u}^{k}\|_G < \eta \|\nabla \bar{u}^{k}_{old}\|_G$?
Yes
No
IR1 = 1?
Yes
No
Yes
Yes
Yes
No
k = ℓ?
Yes

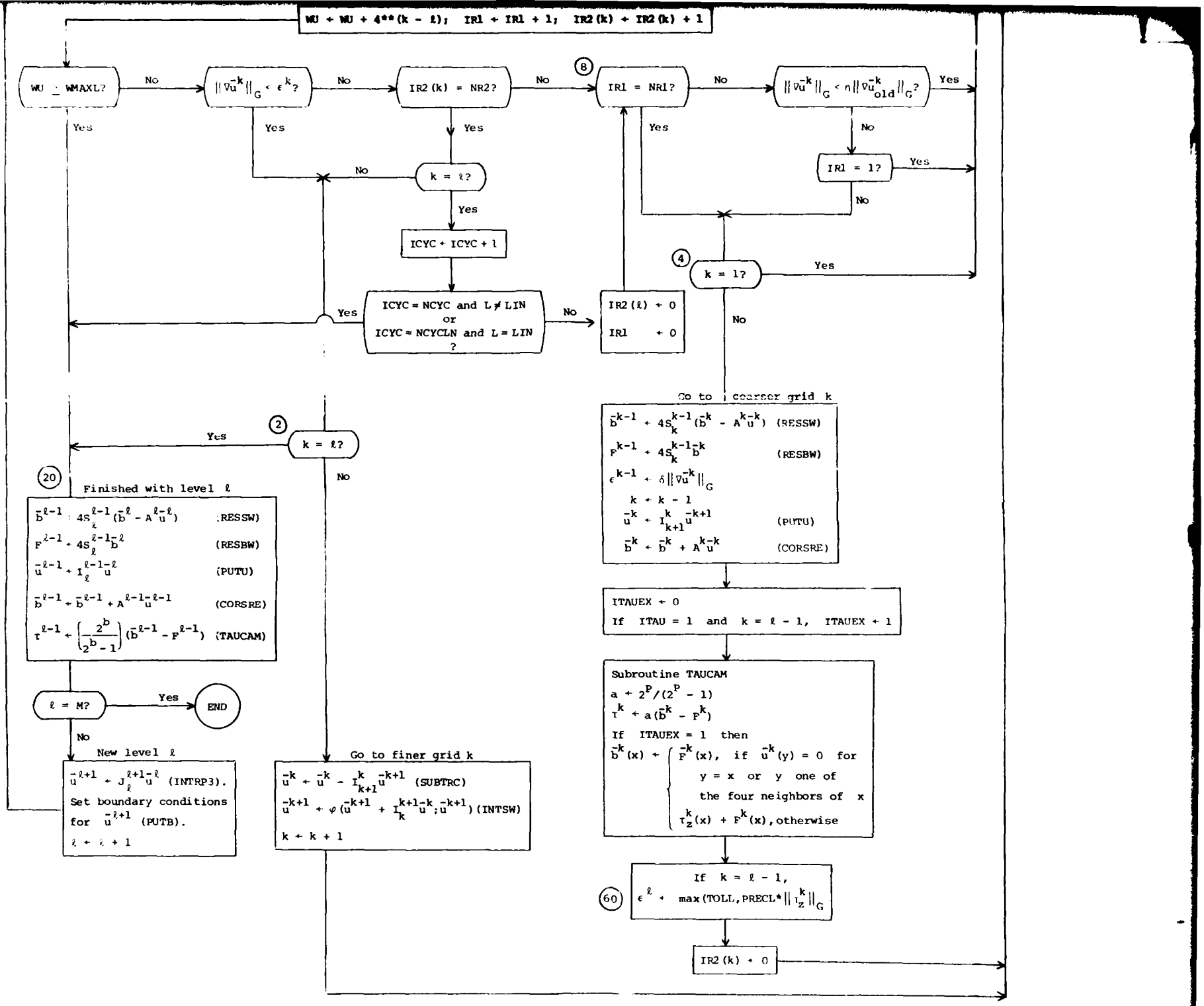


Figure 6.1: Flow Chart for PFMG
(circled numbers are statement numbers)

II. $\bar{u}^k$ is used to estimate the local truncation error on G^{k-1}.

Suppose that the difference approximations are of order p and that $\bar{u}^k$ can be extended to a smooth function on Ω. Then on G^{k-1},

$$A^{k-1}I_k^{k-1}\bar{u}^k \doteq h_{k-1}^2 \mathcal{L}\bar{u}^k + \tau^{k-1} , \tag{6.4}$$

and

$$S_k^{k-1}A^k\bar{u}^k \doteq h_k^2 \mathcal{L}\bar{u}^k + 2^{-(p+2)}\tau^{k-1} , \tag{6.5}$$

where the local truncation error τ^{k-1} depends upon the derivatives of $\bar{u}^k$. Eliminating the unknown $\mathcal{L}\bar{u}^k$ we obtain

$$\tau^{k-1} \doteq \frac{2^p}{2^p - 1}\left[A^{k-1}I_k^{k-1}\bar{u}^k - 4S_k^{k-1}A^k\bar{u}^k\right] , \tag{6.6}$$

$$= \frac{2^p}{2^p - 1}\left[\{4S_k^{k-1}(\bar{b}^k - A^k\bar{u}^k)\} + \{A^{k-1}I_k^{k-1}\bar{u}^k\} - \{4S_k^{k-1}\bar{b}^k\}\right] . \tag{6.7}$$

In PFMG, the first { } in (6.7) is evaluated in subroutine RESSW; the second { } is computed and added to the first using subroutines CORSRE and PUTU; the third { } is evaluated in subroutine RESBW (which is a minor modification of RESSW); and, finally, τ^{k-1} is estimated in subroutine TAUCAM. The estimate (6.7) is not accurate near the discrete interface, and so TAUCAM computes τ_z^{k-1} where

$$\tau_z^{k-1}(x) = \begin{cases} \tau^{k-1}(x), & \text{if } \bar{u}^{k-1}(x) > 0 \\ 0 , & \text{if } \bar{u}^{k-1}(x) = 0 . \end{cases} \tag{6.8}$$

Because of the lack of smoothness of the solution near the free boundary, it is not entirely clear what the value of p should be. It is known (Brezzi and Sacchi [1976]) that the convergence of the finite difference approximations is probably only $O(h^1)$ in the $W^{1,2}(\Omega)$ norm, and Nitsche [1975] has proved $O(h^2 \ell n\, h)$ convergence in the infinity norm. However, these are global error bounds, while we are concerned with the asymptotic behavior of the local truncation error τ. Except in a neighborhood of the discrete interface Γ^ℓ, p is clearly equal to 2. Since the choice of p may vary over Ω, we could perhaps set $p = 1$ near Γ^ℓ, but the values of τ near Γ^ℓ are not very accurate and so, for simplicity, we have taken $p = 2$ everywhere.

III. As usual in numerical analysis the estimate (6.7) for τ^{k-1} can be used in two ways:

(a) To estimate the error $\bar{u}^k - u$.

Since $\tau^k \doteq 2^{-2-p}\tau^{k-1}$, and remembering that G^k has four times as many points as G^{k-1} but $h_{k-1} = 2h_k$, we see from (2.23) that

$$\|\tau_z^k\|_G \doteq \|\tau_z^{k-1}\|_G/2^p . \tag{6.9}$$

Combining (6.7), (6.8) and (6.9) we obtain an estimate for $\|\tau_z^k\|_G$.

In the previous sections we were concerned with asymptotic convergence. That is, we were concerned with the rate of convergence of $\bar{u}^k$ to $\bar{U}^k$ over a very large number of iterations. However, if we want an approximation to the solution u of (1.1), it is only necessary to iterate until the residual on G is small compared with the truncation error, that is, until

$$\|\nabla\bar{u}^k\|_G \doteq 0(\|\tau_z^k\|_G) . \tag{6.10}$$

Once (6.10) holds, further computation will improve the accuracy of $\bar{u}^k$ as a solution of the finite difference equations but will not improve its accuracy as an approximation to u. Noting (6.9), we see that (6.10) will certainly be true if

$$\|\nabla\bar{u}^k\|_G \leq \|\tau_z^{k-1}\|_G . \tag{6.11}$$

The stopping criterion (6.11) is incorporated in PFMG by setting

$$\epsilon^\ell = \max\{\text{PRECL}*\|\tau_z^{\ell-1}\|_G, \text{TOL}*\text{RATIO}**\text{L}\} \tag{6.12}$$

where PRECL, TOL, and RATIO are input parameters. (If TOL = 0, RATIO = 1, and PRECL = 1 then (6.12) reduces to (6.11) for $k = \ell$).

(b) Improvement of accuracy of $\bar{u}^{k-1}$.

Once an estimate for the truncation error τ_z^{k-1} is available, it can be used to improve the accuracy of the difference approximation on G^{k-1} by replacing $F^{k-1}(x)$ by $F^{k-1}(x) + \tau_z^{k-1}(x)$ (see (6.4)). This is only done at points $x \in G^{k-1}$ such that $\bar{u}^{k-1}(y) > 0$ for all four neighbors $y \in G^{k-1}$ of x since the value of τ_z^{k-1} is not

accurate elsewhere. In PFMG this is done in the subroutine TAUCAM when $k = \ell - 1$ and the input parameter ITAU = 1.

Of course, this is only meaningful when $\|\tau_z^{k-1}\|_G$ is small compared to $\|\nabla u^{-k}\|_G$: if the iterations are continued for a long time then convergence will not occur because the conditions of Lemma 5.1 will be violated, but PFMG is never used in this way. In fact, experience with equalities indicates that when τ-extrapolation is used, the best procedure is to avoid relaxation after returning for the last time to the finest grid.

IV. As already mentioned, the logic of PFMG is more complicated than that of PFAS; it is best understood by consulting Figure 6.1 and Appendix C. Several parameters are introduced and this enables one to control explicitly the number of G^k projected sweeps at any level k, and the number of cycles at level ℓ. If

$$NR1 = NR2 = NCYC = NCYCM = -1 ,$$

$$PREC = PRECM = 0 ,$$

$$RATIO = 1 ,$$

then the logic of PFMG reduces to that of PFAS.

We now describe numerical results obtained using PFMG to solve the dam problem (4.1), (4.2). In all cases, G^1 is a $(2 + 1) \times (3 + 1)$ grid and LIN = 2.

To control the iterations we set NR1 = 2, NR2 = 3, NCYC = 1, NCYCLN = 3, and NCYCM = 10. The result is that in each cycle on grid G^ℓ, two G^k projected sweeps are carried out for $1 < k \leq \ell$ as we descend from G^ℓ to G^1, and one G^k projected sweep is carried out as we ascend from G^1 to G^ℓ. For $\ell = LIN$ up to three G^ℓ cycles are allowed, so that a good initial approximation can be obtained. For $LIN < \ell < M$ only one G^ℓ cycle is allowed, while up to 10 G^M cycles are allowed. This will be clearer after consulting Figure 6.2 which shows the output for $M = 4$.

```
.................................................. 2 ...................
LEVEL 2   RESIDUAL NORM=  .266+001   WORK=  1.000 IR1=  1 IR2(K)= 1
LEVEL 2   RESIDUAL NORM=  .174+001   WORK=  2.000 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .920+000     K= 1
LEVEL 1   RESIDUAL NORM=  .803+000   WORK=  2.250 IR1=  1 IR2(K)= 1
LEVEL 1   RESIDUAL NORM=  .130+000   WORK=  2.500 IR1=  2 IR2(K)= 2
LEVEL 1   RESIDUAL NORM=  .814-002   WORK=  2.750 IR1=  3 IR2(K)= 3
LEVEL 2   RESIDUAL NORM=  .889+000   WORK=  3.750 IR1=  1 IR2(K)= 3
LEVEL 2   RESIDUAL NORM=  .258+000   WORK=  4.750 IR1=  1 IR2(K)= 1
LEVEL 2   RESIDUAL NORM=  .102+000   WORK=  5.750 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .924+000     K= 1
LEVEL 1   RESIDUAL NORM=  .238-001   WORK=  6.000 IR1=  1 IR2(K)= 1
LEVEL 1   RESIDUAL NORM=  .149-002   WORK=  6.250 IR1=  2 IR2(K)= 2
LEVEL 1   RESIDUAL NORM=  .930-004   WORK=  6.500 IR1=  3 IR2(K)= 3
LEVEL 2   RESIDUAL NORM=  .484-001   WORK=  7.500 IR1=  1 IR2(K)= 3
LEVEL 2   RESIDUAL NORM=  .157-001   WORK=  8.500 IR1=  1 IR2(K)= 1
LEVEL 2   RESIDUAL NORM=  .443-002   WORK=  9.500 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .983+000     K= 1
LEVEL 1   RESIDUAL NORM=  .956-003   WORK=  9.750 IR1=  1 IR2(K)= 1
LEVEL 1   RESIDUAL NORM=  .597-004   WORK= 10.000 IR1=  2 IR2(K)= 2
LEVEL 1   RESIDUAL NORM=  .373-005   WORK= 10.250 IR1=  3 IR2(K)= 3
LEVEL 2   RESIDUAL NORM=  .117-002   WORK= 11.250 IR1=  1 IR2(K)= 3
                                                GREEN NORM OF TAU-Z =    .983+000     K= 1
          SOLUTION ERROR: L INFINITY NORM =   .60769+000     GNORM =     .19669+000
          SOLUTION      : L INFINITY NORM =   .28800+003     GNORM =     .12949+003
          RELATIVE ERROR: L INFINITY NORM =   .21100-002     GNORM =     .15189-002
.................................................. 3 ...................................
LEVEL 3   RESIDUAL NORM=  .946+000   WORK=  3.812 IR1=  1 IR2(K)= 1
LEVEL 3   RESIDUAL NORM=  .265+000   WORK=  4.812 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .114+001     K= 2
LEVEL 2   RESIDUAL NORM=  .696-001   WORK=  5.062 IR1=  1 IR2(K)= 1
LEVEL 2   RESIDUAL NORM=  .257-001   WORK=  5.313 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .140+001     K= 1
LEVEL 1   RESIDUAL NORM=  .138-001   WORK=  5.375 IR1=  1 IR2(K)= 1
LEVEL 1   RESIDUAL NORM=  .143-002   WORK=  5.437 IR1=  2 IR2(K)= 2
LEVEL 1   RESIDUAL NORM=  .894-004   WORK=  5.500 IR1=  3 IR2(K)= 3
LEVEL 2   RESIDUAL NORM=  .115-001   WORK=  5.750 IR1=  1 IR2(K)= 3
LEVEL 3   RESIDUAL NORM=  .157+000   WORK=  6.750 IR1=  1 IR2(K)= 3
                                                GREEN NORM OF TAU-Z =    .125+001     K= 2
          SOLUTION ERROR: L INFINITY NORM =   .32441+000     GNORM =     .47357+000
          SOLUTION      : L INFINITY NORM =   .28800+003     GNORM =     .43262+003
          RELATIVE ERROR: L INFINITY NORM =   .11264-002     GNORM =     .10947-002
.................................................. 4 ...................................
LEVEL 4   RESIDUAL NORM=  .609+000   WORK=  2.687 IR1=  1 IR2(K)= 1
LEVEL 4   RESIDUAL NORM=  .189+000   WORK=  3.687 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .646+000     K= 3
LEVEL 3   RESIDUAL NORM=  .953-001   WORK=  3.937 IR1=  1 IR2(K)= 1
LEVEL 3   RESIDUAL NORM=  .637-001   WORK=  4.187 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .114+001     K= 2
LEVEL 2   RESIDUAL NORM=  .351-001   WORK=  4.250 IR1=  1 IR2(K)= 1
LEVEL 2   RESIDUAL NORM=  .194-001   WORK=  4.312 IR1=  2 IR2(K)= 2
                                                GREEN NORM OF TAU-Z =    .108+001     K= 1
LEVEL 1   RESIDUAL NORM=  .913-002   WORK=  4.328 IR1=  1 IR2(K)= 1
LEVEL 1   RESIDUAL NORM=  .984-003   WORK=  4.344 IR1=  2 IR2(K)= 2
LEVEL 1   RESIDUAL NORM=  .615-004   WORK=  4.359 IR1=  3 IR2(K)= 3
LEVEL 2   RESIDUAL NORM=  .942-002   WORK=  4.422 IR1=  1 IR2(K)= 3
LEVEL 3   RESIDUAL NORM=  .369-001   WORK=  4.672 IR1=  1 IR2(K)= 3
LEVEL 4   RESIDUAL NORM=  .134+000   WORK=  5.672 IR1=  1 IR2(K)= 3
                                                GREEN NORM OF TAU-Z =    .860+000     K= 3
          SOLUTION ERROR: L INFINITY NORM =   .48932-001     GNORM =     .23107+000
          SOLUTION      : L INFINITY NORM =   .28800+003     GNORM =     .15697+004
          RELATIVE ERROR: L INFINITY NORM =   .16990-003     GNORM =     .14721-003
**** TIME AT ELAPSE IS          .1350 SECONDS *****
```

Figure 6.2: Typical output for the PFMG algorithm
(M = 6, Dam Problem, Run #X67705)

Before discussing how the error was controlled, it is necessary to distinguish between the goals of PFMG and PFAS. Asymptotically, PFMG and PFAS behave the same, because once PFMG has reached level M it performs essentially like PFAS. The purpose of PFMG is to obtain quickly an approximation $\bar{u}^M$ which satisfies the stopping criterion (6.11), namely

$$\|\nabla\bar{u}^M\|_G \leq \|\tau_z^{M-1}\|_G .$$

To achieve this we set

$$\text{PRECM} = 1, \quad \text{TOL} = 0, \quad \text{ETA} = 10 ,$$

$$\text{DELTA} = 0, \quad \text{PREC} = 0, \quad \text{RATIO} = 1 .$$

Finally, we set WMAX = 30, and WMAXM = 40, though these values were of course never reached.

PFMG includes the option of computing, $\|\bar{u}^\ell - u\|_\infty$ and $\|\bar{u}^\ell - u\|_G$, where u is the exact solution. For the dam problem, it is possible to compute u analytically using elliptic integrals (Cryer [1976]) but this has not yet been done: we therefore took u to be the most accurate approximation known to us, namely the approximation $\bar{u}^7$ computed in double precision on a $(128 + 1) \times (192 + 1)$ grid as described in Section 4. For problem (5.3), (5.4) the exact solution is given by (5.4).

We first performed a number of experiments with $M = 2,3,4,$ and 5:

1. τ-extrapolation (with $p = 2$) gave slightly worse results for the dam problem and problem (5.3), (5.4).

2. In contrast to our experience with PFAS, the use of modification 6 had only a slight effect.

3. It was thought that convergence might be improved by multiplying the difference $\nabla\bar{u}^k(x)$ by h for points x near the free boundary before computing $\|\nabla\bar{u}^k(x)\|_G$. This was implemented as a subroutine RELAX1 but was found to have negligible effect.

All the results given below are for the case of no τ-extrapolation (ITAU = 0) and no modification (NINTSW = NRESSW = 1).

The results for the dam problem for different values of M are shown in Table 6.1.

M	2	3	4	5
G^M Work Units	3.75	6.75	5.67	5.41
Execution Time (seconds)	.009	.053	.131	.349
$\|\bar{u}^M - \bar{u}^7\|_\infty / \|u\|_\infty$	.00374	.00112	.000169	.0000623
$\|\bar{u}^M - \bar{u}^7\|_G / \|u\|_G$	.00334	.00109	.000147	.0000405
$\|\nabla\bar{u}^M\|_G$	.889	.157	.134	.0714
$\|\tau_z^{M-1}\|_G$	2.39	1.25	0.86	0.60

Table 6.1: Solution of the dam problem using PFMG.
(Run #X67247)

Since we only have estimates for τ^{M-1}, it is not possible to obtain rigorous error bounds. Nevertheless, it is interesting to apply the error bounds of Section 2.

Let $\tilde{U}^M$ denote the vector obtained by evaluating the solution $u(x)$ on G^M. Then, from (6.4), (1.1), (2.2), (2.3), (2.13), and (3.1),

$$A^M\tilde{U}^M \leq b^M + \tau^M ,$$

so that, from Lemma 2.1,

$$\|\tilde{U}^M - U^M\|_2 \leq \frac{1}{\alpha_M}\|\tau_+^M\|_2 . \tag{6.13}$$

On the other hand, from Lemma 2.2,

$$\|U^M - \bar{u}^M\|_2 \leq \frac{1}{\alpha_M}\|P^M\|_2\|\nabla\bar{u}^M\|_2 .$$

For the dam problem, P is an upper triangular matrix with at most two nonzero elements per row, and $\|P^M\|_2 \leq 2$. Thus,

$$\|U^M - \bar{u}^M\|_2 \leq \frac{2}{\alpha_M}\|\nabla\bar{u}^M\|_2 . \tag{6.14}$$

Combining these inequalities we obtain

$$\|\tilde{U}^M - \bar{u}^M\|_2 \leq \frac{1}{\alpha_M} \left[\|\tau_+^M\|_2 + 2\|\nabla\bar{u}^M\|_2\right] ,$$

or, equivalently,

$$\|\tilde{U}^M - \bar{u}^M\|_G \leq \frac{1}{\alpha_M} \left[\|\tau_+^M\|_G + 2\|\nabla\bar{u}^M\|_G\right] . \tag{6.15}$$

Using (6.8) and (6.9), we conclude that

$$\|\tilde{U}^M - \bar{u}^M\|_G \doteq \frac{1}{\alpha_M} \left[\frac{1}{2^p} \|\tau_z^{M-1}\|_G + 2\|\nabla\bar{u}^M\|_G\right] . \tag{6.16}$$

Next, we note that for the dam problem

$$\alpha_M \doteq \alpha h_M^2 , \tag{6.17}$$

where

$$\alpha = \left(\frac{\pi}{16}\right)^2 + \left(\frac{\pi}{24}\right)^2 \doteq .055 > 14/256 . \tag{6.18}$$

and

$$h_M = 16\ 2^{-M}$$

Thus, finally, for the dam problem,

$$\|\tilde{U}^M - \bar{u}^M\|_G \doteq \frac{2^{2M}}{14} \left[\frac{1}{2^p} \|\tau_z^{M-1}\|_G + 2\|\nabla\bar{u}^M\|_G\right] . \tag{6.19}$$

For example, for $M = 5$ we obtain, using Table 6.1, that

$$\|\tilde{U}^5 - \bar{u}^5\|_G / \|\tilde{U}^5\|_G \doteq \frac{2^{10}}{14} \left[\frac{1}{4}(0.60 + 2(.071)\right] / (5.9\ 10^4) \tag{6.20}$$

$$\doteq .00036 ;$$

the observed value quoted in Table 6.1 is .000040.

In Table 6.2 we repeat the computations of Table 6.1 for the problem (5.3), (5.4).

M	2	3	4	5
G^M Work Units	3.75	6.75	5.672	5.414
Execution Time (seconds)	.028	.103	.263	.842
$\|\bar{u}^M - \tilde{U}^M\|_\infty / \|u\|_\infty$	.0147	.000985	.000266	.0000645
$\|\bar{u}^M - \tilde{U}^m\|_G / \|\tilde{U}^M\|_G$	.0147	.00127	.000376	.0000956
$\|\tau\bar{u}^M\|_G$	10.5	.241	.121	.0764
$\|\tau_z^{M-1}\|_G$	4.18	1.62	1.10	.749

Table 6.2 Solution of problem (5.3), (5.4) using PFMG.

(Run #X67243)

The error estimate (6.19) also holds for the problem (5.3), (5.4), since we are using the Laplace operator on a rectangle with sides in the ratio 2:3. Applying (6.19) we obtain

$$\|\tilde{U}^5 - \bar{u}^5\|_G / \|\tilde{U}^5\|_G \leq \frac{2^{10}}{14} \left| \frac{1}{4}(.75) + 2(.076) \right| / (1.2\ 10^4) ,$$

$$\doteq .0021 ;$$

the observed value quoted in Table 6.2 is .0000645.

The behavior of the global error $\bar{u}^M - u$ can be checked using Tables 6.1 and 6.2. From Table 6.1 we have

$$\left[\frac{\|\bar{u}^5 - \bar{u}^7\|_\infty}{\|\bar{u}^2 - \bar{u}^7\|_\infty} \right]^{1/3} = \left[\frac{.0000623}{.00374} \right]^{1/3} \doteq \frac{1}{2^{1.96}}$$

In Table 6.2 the error in $\bar{u}^2$ is "abnormally large". However,

$$\left[\frac{\|\bar{u}^5 - u\|_\infty}{\|\bar{u}^3 - u\|_\infty} \right]^{1/2} = \left[\frac{.0000645}{.000985} \right]^{1/2} \doteq \frac{1}{2^{1.96}}$$

These results strongly suggest that the global error is $O(h^2)$.

The behavior of the local error τ can also be checked using Tables 6.1 and 6.2. From Table 6.1,

$$[\|\tau_z^4\|_G/\|\tau_z^1\|_G]^{1/3} = [.60/2.39]^{1/3} \doteq 1/2^{.66} ,$$

while, from Table 6.2,

$$[\|\tau_z^4\|_G/\|\tau_z^1\|_G]^{1/3} = [4.18/.749]^{1/3} \doteq 1/2^{.82} ,$$

so that $\tau = 0(h^q)$ with $q \in (.66,.82)$. This explains why τ-extrapolation with $p = 2$ did not reduce the computational effort for these problems. The essential difficulty is of course that the irregularity of the discrete interface makes it difficult to obtain accurate estimates for τ. In fact, τ-extrapolation with $p = 1$ was found to be slower than τ-extrapolation with $p = 2$.

Finally, in Table 6.3 we repeat the computations of Table 5.3 for a tolerance $\epsilon^M = .0714$, the value of $\|\nabla u^{-5}\|_G$ in Table 6.1. We are thus comparing the performance of PFAS (with modification 6), PFMG, and projected SOR for comparable errors.

Method	PFMG	PFASMD (M6)	Projected SOR
Work Units	5.41	9.64	56.0
$\|\nabla u^{-M}\|_G$	.0714	.0239	.0695
Execution Time (seconds)	.349	.440	1.94

Table 6.3: Solution of the dam problem for $M = 5$ and $\epsilon^M = .0714$ using PFASMD (modification 6), PFMG, and projected SOR. (Runs #X67247 and #X67250)

From Table 6.3, we see that PFMG is faster than projected SOR even when only low accuracy is required. PFAS and PFMG require comparable times, but PFMG gives much more information and is, therefore, preferable. PFMG also uses fewer work units than PFAS. This is significant because the number of work units used is independent of

the computer. Furthermore, on the basis of experience with many problems, it can be said that the number of work units used does not vary greatly with the problem: for most operators $\mathcal{L}$ PFMG requires only 5.4 work units.

We conclude this section with some remarks on the implementation of PFMG:

1. From Table 6.3 we see that the execution time per work unit of PFMG is greater than the comparable quantity for PFAS by a factor

$$\frac{.349}{5.41} \Big/ \frac{.440}{9.64} = 1.41 \ .$$

This additional overhead is probably due to the cubic interpolation used by J_{k-1}^{k}, and could perhaps be reduced by better programming. When $\mathcal{L}$ is complicated, the additional overhead required by PFMG is relatively much less significant: it is only with a very simple operator like the 5-point Laplacian that the additional overhead is so expensive.

2. In PFMG one often need not have <u>any</u> storage for the finest grid G^M - not even external storage. The algorithm visits G^M only twice: at the beginning of the last cycle and at the end of the last cycle.

At the beginning of the cycle, the following operations are performed: interpolation (J_{M-1}^{M}); two G^M projected sweeps; and residual transfer (I_M^{M-1} and S_M^{M-1}). All these operations can be made in one passage over G^M, in such a way that only four columns of G^M are held in memory at one time. Each time a new column, say column i, is created (by interpolation), a relaxation can be made in column $i-1$, then the second relaxation can already be made in column $i-2$ and the residuals from column $i-3$ can be transferred back to the coarse grid. Column $i-4$ can simultaneously be discarded (i.e., replaced by column i). After this visit to G^M all the information is available (in $\bar{F}^{M-1}$ and $\bar{u}^{M-1}$) to solve the G^{M-1} problem to the truncation level of G^M.

The final return to G^M (which would require the storage of the previous values of U^M) is made in order to obtain the solution on G^M rather than on G^{M-1}, but it does not improve its pointwise accuracy. If one is only interested in knowing some functionals of the solution, these can be calculated without having the final solution on G^M. To approximate a functional $\mathcal{H}(U)$, for example, one computes $\mathcal{H}(\bar{u}^{M-1}) + c_M^{M-1}$,

where $\sigma_M^{M-1} = \mathcal{H}(\bar{u}^M) - \mathcal{H}(I_M^{M-1}\bar{u}^M)$, $\bar{u}^{M-1}$ is the final solution on G^{M-1}, and $\bar{u}^M$ is the last solution on G^M before switching back to G^{M-1}. Clearly, σ_M^{M-1} can be calculated during the above-mentioned passage on G^M. Note that σ_M^{M-1} is a "relative truncation correction", similar to τ_M^{M-1}. It makes the approximation $\mathcal{H}(\bar{u}^{M-1}) + \sigma_M^{M-1}$ correct to the G^M truncation level. $\mathcal{H}$ need not be a linear functional.

7. CONCLUSIONS AND RECOMMENDATIONS.

1. Multigrid methods can easily be adapted to handle linear complementarity problems arising from free boundary problems (see Table 4.2).

2. Multigrid methods are superior to projected SOR and modified block SOR (see Tables 5.3 and 6.3, and Section 4).

3. For high accuracy solutions of the discrete LCP, one should use PFASMD with modification 6 (see Tables 5.1 and 5.2).

4. For solutions which are accurate to within truncation error one should use PFMG, with no modifications (see Tables 6.1, 6.2, and 6.3).

Finally, we conclude with some comments suggesting possible future applications of multigrid methods to complementarity problems:

1. For equalities, experience has shown that multigrid methods are as efficient for problems where $\mathcal{L}$ is nonlinear as for problems where $\mathcal{L}$ is linear.

2. Experience from equalities indicates that with similar efficiency (just a few more work units) one can solve much more difficult problems, such as problems in which the coefficients of $\mathcal{L}$ vary by orders of magnitude (e.g., large variations in the diffusivity of the dam). In such cases SOR and other methods converge very slowly. See Alcouffe et. al. (to appear).

3. The truncation error near a discrete interface cannot be reduced by using higher order approximations because the second derivatives are usually discontinuous. A good way to improve the approximation would be to use finer mesh sizes near the discrete interface. This can be combined very effectively with the multigrid process (see Brandt [1979, Section 3]). In fact, a vast improvement is expected if τ-extrapolation is used <u>together</u> with local refinements. Fine levels will then be used only near the interface.

Acknowledgement.

We thank Professor R. Sacher for making available a copy of his program for solving LCP's using the modified block SOR algorithm of Cottle and Sacher, and for his comments on an early version of this report.

AB/CWC/ed

REFERENCES

R. ALCOUFFE, A. BRANDT, J. DENDY and J. PAINTER. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. Los Alamos Scientific Laboratory Report, to appear.

C. BAIOCCHI. Sur un probleme a frontiere libre traduisant le filtrage de liquides a travers des milieux poreux. Comptes Rendus Acad. Sci. Paris, A273(1971), pp. 1215-1217.

C. BAIOCCHI. Free boundary problems and variational inequalities. Technical Summary Report No. 1883, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1978.

M. L. BALINSKI and R. W. COTTLE. Complementarity and Fixed Point Problems. North-Holland, Amsterdam, 1978.

J. BEAR. Dynamics of Fluids in Porous Media. American Elsevier, New York, 1972.

A. BRANDT. Multi-level adaptive solutions to boundary value problems. Math. Computation, 31(1977), pp. 333-390.

A. BRANDT. Multi-level adaptive techniques (MLAT) for singular-perturbation problems. In Numerical Analysis of Singular Perturbation Problems, P. W. Hemker and J. J. H. Miller (editors). New York, Academic Press 1979, pp. 53-142.

A. BRANDT and N. DINAR. Multi-grid solutions to elliptic flow problems. Symposium on Numerical Solution of Partial Differential Equations, S. V. Parter, ed. New York, Academic Press, 1979, pp. 53-147.

F. BREZZI and G. SACCHI. A finite element approximation of variational inequalities related to hydraulics. Calcolo, 13(1976), pp. 259-273.

J. CEA, R. GLOWINSKI, and J. C. NEDELEC. Application des methodes d'optimisation, de differences et d'elements finis a l'analyse numerique de la torsion elasto-plastique d'une barre cylindrique. In Approximation et Methodes Iteratives de Resolution d'Inequations Variationelles et de Problemes Non Lineaires. Cahier de l'IRIA, No. 12, 1974, pp. 7-138.

G. CIMATTI. On a problem of the theory of lubrication governed by a variational ineuqality. Applied Math. and Optimization, 3(1977), pp. 227-242.

R. W. COTTLE. Computational experience with large-scale linear complementarity problems. Technical Report No. SOL 74-13, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1974.

R. W. COTTLE, F. GIANNESSI, and J. L. LIONS (editors). Variational Inequalities and Complementarity Problems. John Wiley, New York, 1980.

R. W. COTTLE, G. H. GOLUB, and R. S. SACHER. On the solution of large, structured linear complementarity problems: the block partitioned case. Applied Mathematics and Optimization 4(1978), pp. 347-363.

R. W. COTTLE and R. S. SACHER. On the solution of large, structured linear complementarity problems: the tridiagonal case. Applied Mathematics and Optimization, 3(1977), pp. 321-340.

C. W. CRYER. The solution of a quadratic programming problem using systematic overrelaxation. SIAM J. Control, 9(1971), pp. 385-392.

C. W. CRYER. The method of Christopherson for solving free boundary problems for infinite journal bearings by means of finite differences. Math. Comp., 25(1971a), pp. 435-444.

C. W. CRYER. A survey of steady state porous flow free boundary problems. Technical Summary Report No. 1657, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1976.

C. W. CRYER. A bibliography of free boundary problems. Technical Summary Report No. 1793, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1977.

C. W. CRYER. The solution of the axisymmetric elastic-plastic torsion of a shaft using variational inequalities. Technical Summary Report No. 1948, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1979. To appear in J. Math. Anal. Appl.

C. W. Cryer. Successive overrelaxation methods for solving linear complementarity problems arising from free boundary problems. Proceedings, Seminar on Free Boundary Problems, Pavia, October 1979a, to appear.

G. DUVAUT and J. L. LIONS. Inequalities in Mechanics and Physics. Dunod, Paris, 1976.

R. S. FALK. Error estimates for the approximation of a class of variational inequalities. Math. Computation, 28(1974), pp. 963-971.

R. GLOWINSKI. La methode de relaxation. Rendiconti di Matematica, 14 (1971), 56 pages.

R. GLOWINSKI. Finite elements and variational inequalities. Technical Summary Report No. 1885, Mathematics Research Center, University of Wisconsin, 1978.

R. GLOWINSKI, J. L. LIONS, and R. TREMOLIERES. Analyse Numerique des Inequations Variationnelles. Dunod, Paris, 1976.

D. KINDERLEHRER and G. STAMPACCHIA. An Introduction to Variational Inequalities and their Applications. Academic Press, New York, 1980.

H. LANCHON. Torsion elastoplastique d'un arbre cylindrique de section simplement ou multiplement connexe. J. Mecanique, 13(1974), pp. 267-320.

J. A. NITSCHE. L-infinity convergence of finite element approximations. In Mathematical Aspects of the Finite Element Method, Rome Italy, 1975.

R. S. Varga. Matrix Iterative Analysis. Prentice Hall, Englewood Cliffs, N. J., 1962.

```
======  APPX-A-PFAS  =====

C     ****************************************************************
C
C     THIS PROGRAM SOLVES THE PROBLEM OF POROUS FLOW THROUGH A
C     RECTANGULAR DAM OF HEIGHT Y1 AND WIDTH A.
C     THE RESERVOIR TO THE RIGHT OF THE DAM IS OF HEIGHT Y2.
C
C     WRITTEN BY ACHI BRANDT AND COLIN CRYER AUGUST 1980
C
C     ADDITIONAL PARAMETERS USED ARE:
C     NX0       THE NUMBER OF GRID INTERVALS IN THE X-DIRECTION IN
C               THE COARSEST GRID, GRID 1.
C     NY0       THE NUMBER OF GRID INTERVALS IN THE Y-DIRECTION IN
C               THE COARSEST GRID, GRID 1.
C     H0        THE GRID SIZE IN THE COARSEST GRID, GRID 1.
C     M         THE NUMBER OF GRIDS TO BE USED.
C     TOL       THE TOLERANCE. COMPUTATION TERMINATES IF THE RESIDUAL
C               ON THE FINEST GRID IS LESS THAN TOL.
C     WMAX      THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
C               FINEST GRID. COMPUTATION TERMINATES WHEN WMAX IS EXCEEDED.
C               IN PRACTICAL CASES, ONE SETS WMAX=30. IN THE PRESENT WORK,
C               WE OFTEN SET WMAX=100 SO AS TO OBSERVE THE ASYMPTOTIC
C               BEHAVIOR OF THE ALGORITHM.
C     MPRINT    THE GRID TO BE PRINTED AT THE END OF THE COMPUTATION.
C               THAT IS, WE PRINT THE MPRINT SUBSET OF THE FINAL ANSWER
C               ON THE GRID M.
C     NQSIZE    SIZE OF ARRAY Q
C               MUST BE CHANGED FOR LARGE PROBLEMS BY EDITING PROGRAM
C               =18000 FOR DAM PROBLEM M=2,3,4,5,6
C               =70000 FOR DAM PROBLEM M=7
C
C
C     ALL THE PARAMETERS ARE SET IN THE PROGRAM, BUT THEIR VALUES
C     CAN BE RESET ON THE NAMELIST INPUT CARD WHICH IS READ IN
C     BY THE PROGRAM.
C     THE NAMELIST CARD MUST BE PROVIDED AS INPUT.
C
C     THE PROGRAM SETS UP STORAGE FOR THE SOLUTIONS AND RIGHT
C     HAND SIDES.
C     THE SOLUTIONS ARE STORED IN ARRAYS 1 TO M.
C     THE RIGHT HAND SIDES ( OR, SOMETIMES THE RESIDUALS )
C     ARE STORED IN ARRAYS M+1 TO 2*M.
C
C     THIS PROGRAM WAS USED TO COMPUTE THE RESULTS IN FIGURE 3.2
C     AND TABLES 4.1 AND 4.2 OF THE MRC REPORT.
C
C     ****************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL G,F
      COMMON /PRBDAT/Y1,Y2,A
      COMMON /QDAT/NQSIZE,NQERR
      NAMELIST /INDAT/Y1,Y2,A,NX0,NY0,H0,M,TOL,WMAX,MPRINT
      NQSIZE=18000
      Y1=24
      Y2=4
      A=16
      NX0=4
      NY0=6
```

```
      H0=4.
      M=3
      TOL=0.
      WMAX=30.
      MPRINT=1
      READ(5,INDAT)
      WRITE(6,INDAT)
C     SET TIME TO ZERO
      CALL URTIMS(0.0)
      CALL PFAS(NX0,NY0,H0,M,TOL,WMAX,G,F)
C     PRINT ELAPSED TIME
      T=URTIMG('ELAPSED TIME')
      CALL SOLPRT(M,MPRINT)
      STOP
      END
C
C
      DOUBLE PRECISION FUNCTION F(X,Y)
C     DAM PROBLEM
C     THIS SUBROUTINE COMPUTES THE RIGHT HAND SIDE OF THE
C     GOVERNING POISSON EQUATION DEL*DEL U=F.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      F=1.
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION G(X,Y)
C     DAM PROBLEM
C     THIS SUBROUTINE COMPUTES THE BOUNDARY DATA AND THE
C     INITIAL APPROXIMATION TO THE SOLUTION U.
C     THE INITIAL APPROXIMATION IS OBTAINED BY LINEAR INTERPOLATION
C     IN THE X-DIRECTION BETWEEN THE GIVEN BOUNDARY DATA.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /PRBDAT/Y1,Y2,A
      G1=.5*(Y1-Y)**2
      G2=.5*(Y2-Y)**2
      IF( Y.GE.Y2) G2=0
      G=(G1*(A-X)+ G2*X)/A
      RETURN
      END
C
C
      SUBROUTINE PFAS(NX0,NY0,H0,M,TOL,WMAX,U1,F)
C     THIS SUBROUTINE IS THE MAIN MULTIGRID SUBROUTINE.
C     IT INITIALIZES THE PROBLEM, AND REPEATEDLY CALLS
C     THE SUBROUTINES RELAX,RESCAL,PUTU,CORSRE,SUBTRC,AND INTADD.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /QDAT/NQSIZE,NQERR
      EXTERNAL U1,F
      DIMENSION EPS(10)
C
C
C     SET UP ARRAYS 1 TO M FOR THE SOLUTIONS
C     AND ARRAYS M+1 TO 2*M FOR THE RIGHT HAND SIDES,
C     AND CHECK THAT Q ARRAY IS LARGE ENOUGH
      NQERR=0
```

```
      DO 1 K=1,M
      K2=2**(K-1)
      CALL GRDFN(K,NX0*K2+1,NY0*K2+1,H0/K2)
    1 CALL GRDFN(K+M,NX0*K2+1,NY0*K2+1,H0/K2)
      PRINT 10,NQSIZE
   10    FORMAT(' SIZE OF Q ARRAY = ', I10)
      IF(NQERR.EQ.0)GOTO 12
      PRINT 11,NQERR
   11    FORMAT(' *** ERROR IN GRDFN *** ARRAY Q NOT LARGE ENOUGH ***',
     * /,' ARRAY Q SIZE SHOULD BE AT LEAST =', I10)
      STOP
   12 CONTINUE
C
C
C     INITIALIZE
      EPS(M)=TOL
      K=M
      WU=0
      CALL PUTF(M,U1,0)
      CALL PUTF(2*M,F,2)
      ETA=.5
      DELTA=.15
C
C     START OF MAIN LOOP IN WHICH ONE MODIFIED GAUSS-SEIDEL
C     SWEEP ON GRID K IS MADE.
C
    5 ERR=1.E30
    3 ERRP=ERR
      CALL RELAX(K,K+M,ERR)
      IF (WU .LE. 0) ERRBEG=ERR
      WU=WU+4.**(K-M)
      WRITE(6,4)K,ERR,WU
    4 FORMAT(' LEVEL',I2,'   RESIDUAL NORM=', D10.3,'   WORK=', F7.3)
      IF(ERR.LT.EPS(K))GOTO 2
      IF (WU.GE.WMAX)RETURN
      IF(K.EQ.1.OR.ERR/ERRP.LT. ETA)GO TO 3
C
C     GO TO COARSER GRID
      IF( K.NE.M .OR. WU.LE.3 ) GOTO 92
      FMU=0.0
      IF( ERR.GT.0 ) FMU=(ERR/ERRBEG)**(1.D0/(WU-1))
      PRINT 91,FMU
   91 FORMAT(' ', 20('*'),'END OF CYCLE',20('*'),'MU = ',F8.4)
   92 CONTINUE
      CALL RESCAL(K,K+M,K+M-1)
      EPS(K-1)=DELTA*ERR
      K=K-1
      CALL PUTU(K+1,K)
      CALL CORSRE(K,K+M)
      GOTO 5
C
C     GO TO FINER GRID
    2 IF (K.EQ.M)RETURN
      CALL SUBTRC(K+1,K)
      CALL INTADD(K,K+1)
      K=K+1
      GOTO 5
```

```
      END
C
C
      SUBROUTINE CORSRE(K,KRHS)
C      APPLIES THE DIFFERENCE OPERATOR ON GRID K
C      TO THE GRID FUNCTION IN ARRAY K, AND ADDS THE RESULT TO THE
C      VALUES IN ARRAY KRHS.
C       KRHS    KRHS      K  K,0
C      B     = R       + A  U
C
C      THE RESULT IS STORED IN ARRAY KRHS.
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(200),IRHS(200)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      DO 1 I=2,I1
      IR=IRHS(I)
      IO=IST(I)
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
      A=-Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
    1 Q(IR+J)=-A-4.*Q(IO+J)
      RETURN
      END
C
C
      SUBROUTINE GRDFN(N,IMAX,JMAX,HH)
C      SETS UP ARRAY N.
C      IMAX    THE DIMENSION IN THE X DIRECTION
C      JMAX    THE DIMENSION IN THE Y DIRECTION
C      HH      THE GRID SIZE
C      THE ARRAY NST CONTAINS THE STARTING ADDRESSES OF THE ARRAYS.
C      THE ARRAY IMX CONTAINS THE MAXIMUM ROW NUMBERS
C      THE ARRAY JMX CONTAINS THE MAXIMUM COL NUMBERS
C      THE ARRAY H   CONTAINS THE GRID SIZES.
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
       COMMON /QDAT/NQSIZE,NQERR
      DATA IQ/1/
      NST(N)=IQ
      IMX(N)=IMAX
      JMX(N)=JMAX
      H(N)=HH
      IQ=IQ+IMAX*JMAX
       IF(IQ.LE.NQSIZE+1) RETURN
       NQERR=IQ-1
      END
C
C
      SUBROUTINE INTADD(KC,KF)
C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
C      AND ADDS TO SOLUTION ON GRID  KF.
C       KF         KF  KC    KF    KF
```

```
      C      U   = PHI( I   W   + U   ; U  )
      C                KC
      C
             IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON Q(18000),ISTC(200),ISTF(200)
            CALL KEY(KC,ISTC,IIC,JJC,HC)
            CALL KEY(KF,ISTF,IIF,JJF,HF)
            DO 1 IC=2,IIC
            IF=2*IC-1
            JF=1
            IFO=ISTF(IF)
            IFM=ISTF(IF-1)
            ICO=ISTC(IC)
            ICM=ISTC(IC-1)
            DO 1 JC=2,JJC
            JF=JF+2
            A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
            AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
            Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
            Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
            Q(IFO+JF-1)=Q(IFO+JF-1)+A
          1 Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
            RETURN
            END
      C
      C
            SUBROUTINE KEY(K,IST,IMAX,JMAX,HH)
      C      RECOVERS THE INFORMATION ABOUT ARRAY K SET UP BY
      C      THE SUBROUTINE GRDFN.
      C      THE VALUE OF THE GRID FUNCTION AT THE POINT (I,J)
      C      IS ADDRESSED AS U(IST(J)+I).
      C
             IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
            DIMENSION IST(1)
            IMAX=IMX(K)
            JMAX=JMX(K)
            IS=NST(K)-JMAX-1
            DO 1 I=1,IMAX
            IS=IS + JMAX
          1 IST(I)=IS
            HH=H(K)
            RETURN
            END
      C
      C
            SUBROUTINE PUTF(K,F,NH)
      C      INSERTS THE VALUES OF THE FUNCTION F
      C      EVALUATED AT THE POINTS OF GRID K
      C      AND MULTIPLIED BY GRIDSIZE**NH
      C      INTO THE ARRAY K.
      C
             IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON Q(18000),IST(600)
            CALL KEY (K,IST,II,JJ,H)
            H2=H**NH
            DO 1 I=1,II
```

```
      DO 1 J=1,JJ
      X=(I-1)*H
      Y=(J-1)*H
    1 Q(IST(I)+J)=F(X,Y)*H2
      RETURN
      END
C
C
      SUBROUTINE PUTU(KF,KC)
C      THIS SUBROUTINE INJECTS THE SOLUTION ON THE FINE GRID
C      KF INTO THE COARSE GRID KC.
C       KC,0    KC  KF
C      U     = I   U
C               KF
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      Q(ICO+JC)=          Q(IFO+JF)
    1 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE    RELAX(K,KRHS,ERR)
C      CARRIES OUT ONE MODIFIED GAUSS-SEIDEL
C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(200),IRHS(200)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      ERR=0.
      DO 1 I=2,I1
      IR=IRHS(I)
      IO=IST(I)
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
       QT=-.25*A
       QN=MAX(0.0,QT)
      ERR=ERR+(QN-Q(IO+J))**2
    1 Q(IO+J)=QN
      ERR=SQRT(ERR)/H
      RETURN
```

```
      END
C
C
      SUBROUTINE RESCAL(KF,KRF,KRC)
C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
C      BEFORE INJECTION, THE RESIDUAL IS SCALED
C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
C      GRIDSIZE ON GRID KC.
C       KRC     KC    KRF   KF  KF
C      R   = 4*S   ( B   - A   U   )
C               KF
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KRF,IRF,IIF,JJF,HF)
      CALL KEY(KRC,IRC,IIC,JJC,HC)
      IIC1=IIC-1
      JJC1=JJC-1
      DO 1 IC=2,IIC1
      ICR=IRC(IC)
      IF=2*IC-1
      JF=1
      IFR=IRF(IF)
      IFO=IUF(IF)
      IFM=IUF(IF-1)
      IFP=IUF(IF+1)
      DO 1 JC=2,JJC1
      JF=JF+2
      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
    1 Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
      RETURN
      END
C
C
       SUBROUTINE SOLPRT(M,MPRINT)
C      PRINTS THE ARRAY M ON THE SUBARRAY MPRINT.
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(600)
       DIMENSION QTEM(100)
      CALL KEY (M,IST,II,JJ,H)
       INTERV=2**(M-MPRINT)
      DO 20 J=JJ,1,-INTERV
       L=0
      DO 10 I=1,II,INTERV
C      X AND Y ARE NOT PRINTED HERE, BUT ARE COMPUTED IN
C      CASE A LATER VERSION NEEDS THEM.
      X=(I-1)*H
      Y=(J-1)*H
       L=L+1
       QTEM(L)=Q(IST(I)+J)
   10     CONTINUE
       PRINT *,(QTEM(LL),LL=1,L)
   20     CONTINUE
```

```
      RETURN
      END
C
C
      SUBROUTINE SUBTRC(KF,KC)
C      THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
C      GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
C       KC    KC    KC  KF
C      W   = U   - I   U
C                  KF
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      Q(ICO+JC)=Q(ICO+JC)-Q(IFO+JF)
    1 CONTINUE
      RETURN
      END
C
C
```

```
      =====  APX-B-PFASMD  =====

 1.     C     ****************************************************************
 2.     C
 3.     C     THIS PROGRAM SOLVES THE PROBLEM OF POROUS FLOW THROUGH A
 4.     C     RECTANGULAR DAM OF HEIGHT Y1 AND WIDTH A.
 5.     C     THE RESERVOIR TO THE RIGHT OF THE DAM IS OF HEIGHT Y2.
 6.     C
 7.     C     WRITTEN BY ACHI BRANDT AND COLIN CRYER AUGUST 1980
 8.     C
 9.     C     THIS PROGRAM WAS USED TO COMPUTE THE RESULTS IN
10.     C     SECTION 5 AND TABLE 6.4 OF THE MRC REPORT.
11.     C
12.     C     ADDITIONAL PARAMETERS USED ARE:
13.     C     NX0      THE NUMBER OF GRID INTERVALS IN THE X-DIRECTION IN
14.     C              THE COARSEST GRID, GRID 1.
15.     C     NY0      THE NUMBER OF GRID INTERVALS IN THE Y-DIRECTION IN
16.     C              THE COARSEST GRID, GRID 1.
17.     C     H0       THE GRID SIZE IN THE COARSEST GRID, GRID 1.
18.     C     M        THE NUMBER OF GRIDS TO BE USED.
19.     C     TOL      THE TOLERANCE. COMPUTATION TERMINATES IF THE RESIDUAL
20.     C              ON THE FINEST GRID IS LESS THAN TOL.
21.     C     WMAX     THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
22.     C              FINEST GRID. COMPUTATION TERMINATES WHEN WMAX IS EXCEEDED.
23.     C              IN PRACTICAL CASES, ONE SETS WMAX=30. IN THE PRESENT WORK,
24.     C              WE OFTEN SET WMAX=100 SO AS TO OBSERVE THE ASYMPTOTIC
25.     C              BEHAVIOR OF THE ALGORITHM.
26.     C     MPRINT   THE GRID TO BE PRINTED AT THE END OF THE COMPUTATION.
27.     C              THAT IS, WE PRINT THE MPRINT SUBSET OF THE FINAL ANSWER
28.     C              ON THE GRID M.
29.     C     NQSIZE   SIZE OF ARRAY Q
30.     C              MUST BE CHANGED FOR LARGE PROBLEMS BY EDITING PROGRAM
31.     C              =18000 FOR DAM PROBLEM M=2,3,4,5,6
32.     C              =70000 FOR DAM PROBLEM M=7
33.     C
34.     C     SWITCHES
35.     C
36.     C     NFGSW    =1 DAM PROBLEM
37.     C              =2 PROBLEM (5.3),(5.4).
38.     C
39.     C
40.     C     NINTSW   =1 INJECTION. SUBROUTINE INTADD
41.     C              =2 MODIFICATION #6. SUBROUTINE INTADM
42.     C                 CORRECTION ONLY ADDED WHEN U.NE.0. SEE (5.15).
43.     C              =3 MODIFICATION #1. SUBROUTINE INTAPR
44.     C                 PHI=MAX(0,U)
45.     C
46.     C     NPUTSW   =1 INJECTION. SUBROUTINES PUTU AND SUBTRC
47.     C              =2 MODIFICATION #2. SUBROUTINES PUTUNN AND SUBTNN.
48.     C                 TRANSFER 0 IF ANY NEIGHBOR ZERO.
49.     C
50.     C     NRELSW   =1 NORMAL RELAXATION. SUBROUTINE RELAX
51.     C              =2 MODIFICATION #3. SUBROUTINE RELXFR
52.     C                 VALUES OF U CHANGED ON GRID
53.     C                 K<M ONLY IF U>0 ON GRID M.
54.     C
55.     C     NRESSW   =1 INJECTION. SUBROUTINE RESCAL
56.     C              =2 MODIFICATION #5. SUBROUTINE RESCL1
57.     C                 USES WEIGHTED RESIDUALS NEAR BOUNDARY.
```

```
C                  RESIDUALS WITH U<0 SET EQUAL TO ZERO
C                =3 MODIFICATION #4. SUBROUTINE RESCAV
C                  USES WEIGHTED RESIDUALS .
C
C     ALL THE PARAMETERS ARE SET IN THE PROGRAM, BUT THEIR VALUES
C     CAN BE RESET ON THE NAMELIST INPUT CARD WHICH IS READ IN
C     BY THE PROGRAM.
C     THE NAMELIST CARD MUST BE PROVIDED AS INPUT.
C
C     THE PROGRAM SETS UP STORAGE FOR THE SOLUTIONS AND RIGHT
C     HAND SIDES.
C     THE SOLUTIONS ARE STORED IN ARRAYS 1 TO M.
C     THE RIGHT HAND SIDES ( OR, SOMETIMES THE RESIDUALS )
C     ARE STORED IN ARRAYS M+1 TO 2*M.
C
C     ************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL G,F
      COMMON /PRBDAT/Y1,Y2,A,R
      COMMON /QDAT/NQSIZE,NQERR
      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
      NAMELIST /INDAT/Y1,Y2,A,R,NX0,NY0,H0,M,TOL,WMAX,MPRINT
     * ,NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
      CHARACTER ITITLE(80)
C
C     READ IN AND PRINT TITLE CARDS
C     FINISH READING TITLE WHEN LAST CARD IS BLANK
C     FINISH RUN WHEN TITLE CARD IS BLANK
      NC=0
    5 READ 10,(ITITLE(I),I=1,80)
   10 FORMAT(80A1)
      NC=NC+1
      PRINT 11,(ITITLE(I),I=1,80)
   11 FORMAT(1H ,80A1)
      DO 12 I=1,80
      IF (ITITLE(I).NE.' ')GOTO 5
   12 CONTINUE
      IF(NC.EQ.1) STOP
C
      NQSIZE=18000
      NFGSW=1
      NINTSW=1
      NPUTSW=1
      NRELSW=1
      NRESSW=1
      Y1=24
      Y2=4
      A=16
      R=32.D0/15.D0
      NX0=4
      NY0=6
      H0=4.
      M=3
      TOL=2.D-8
      WMAX=30.
      MPRINT=1
      READ(5,INDAT)
```

```
      WRITE(6,INDAT)
C     PRINT MODIFICATION NUMBERS
      PRINT 100
100   FORMAT( '0  *** THE FOLLOWING MODIFICATIONS WERE USED *** '/)
      IF(NINTSW.EQ.2) PRINT 106
      IF(NINTSW.EQ.3) PRINT 101
      IF(NPUTSW.EQ.2) PRINT 102
      IF(NRELSW.EQ.2) PRINT 103
      IF(NRESSW.EQ.2) PRINT 105
      IF(NRESSW.EQ.3) PRINT 104
101   FORMAT('0', 'MODIFICATION NUMBER 1')
102   FORMAT('0', 'MODIFICATION NUMBER 2')
103   FORMAT('0', 'MODIFICATION NUMBER 3')
104   FORMAT('0', 'MODIFICATION NUMBER 4')
105   FORMAT('0', 'MODIFICATION NUMBER 5')
106   FORMAT('0', 'MODIFICATION NUMBER 6')
      PRINT 110
110   FORMAT( ' ************* ')
C     SET TIME TO ZERO
      CALL URTIMS(0.0)
      CALL PFASMD(NX0,NY0,H0,M,TOL,WMAX,G,F)
C     PRINT ELAPSED TIME
      T=URTIMG('ELAPSED TIME')
      CALL SOLPRT(M,MPRINT)
      STOP
      END
C
C
      DOUBLE PRECISION FUNCTION F(X,Y)
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       COMMON /PRBDAT/Y1,Y2,A,R
       COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
C      THIS SUBROUTINE COMPUTES THE RIGHT HAND SIDE OF THE
C      GOVERNING POISSON EQUATION DEL*DEL U=F.
       GOTO( 1,2),NFGSW
C
C      DAM PROBLEM
    1  CONTINUE
       F=1.
      RETURN
C
C      PROBLEM OF SECTION 5: (5.3) AND (5.4)
    2  CONTINUE
       D=2.5*R
       A=DMAX1(0.D0,D-R*X-Y)
       B=X+Y
       C=2*(R**2+1)
       F=(C-2.*A*A)*DCOS(B) +4*(R+1)*A*DSIN(B)+2*C
       RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION G(X,Y)
C      THIS SUBROUTINE COMPUTES THE BOUNDARY DATA AND THE
C      INITIAL APPROXIMATION TO THE SOLUTION U.
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       COMMON /PRBDAT/Y1,Y2,A,R
```

```
      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
      GOTO( 1,2),NFGSW
C
C     DAM PROBLEM
C     THE INITIAL APPROXIMATION IS OBTAINED BY LINEAR INTERPOLATION
C     IN THE X-DIRECTION BETWEEN THE GIVEN BOUNDARY DATA.
    1 CONTINUE
      G1=.5*(Y1-Y)**2
      G2=.5*(Y2-Y)**2
      IF( Y.GE.Y2) G2=0
      G=(G1*(A-X)+ G2*X)/A
      RETURN
C
C     PROBLEM OF SECTION 5: (5.3) AND (5.4)
C     INITIAL APPROXIMATION IS A PERTURBATION OF EXACT SOLUTION
    2 CONTINUE
      D=2.5*R
      A=DMAX1(0.D0,D-R*X-Y)
      B=X+Y
      G=A*A*(DCOS(B)+2)
      G=G+X*(3-X)*Y*(2-Y)*10
      RETURN
      END
C
C
      SUBROUTINE PFASMD(NX0,NY0,H0,M,TOL,WMAX,U1,F)
C     THIS SUBROUTINE IS THE MAIN MULTIGRID SUBROUTINE.
C     IT INITIALIZES THE PROBLEM, AND REPEATEDLY CALLS
C     THE SUBROUTINES RELAX,RESCAL,PUTU,CORSRE,SUBTRC,AND INTADD.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /QDAT/NQSIZE,NQERR
      EXTERNAL U1,F
      DIMENSION EPS(10)
C
C
C     SET UP ARRAYS 1 TO M FOR THE SOLUTIONS
C     AND ARRAYS M+1 TO 2*M FOR THE RIGHT HAND SIDES,
C     AND CHECK THAT Q ARRAY IS LARGE ENOUGH
      NQERR=0
      DO 1 K=1,M
      K2=2**(K-1)
      CALL GRDFN(K,NX0*K2+1,NY0*K2+1,H0/K2)
    1 CALL GRDFN(K+M,NX0*K2+1,NY0*K2+1,H0/K2)
      PRINT 10,NQSIZE
   10    FORMAT(' SIZE OF Q ARRAY = ', I10)
      IF(NQERR.EQ.0)GOTO 12
      PRINT 11,NQERR
   11    FORMAT(' *** ERROR IN GRDFN *** ARRAY Q NOT LARGE ENOUGH ***',
     * /,' ARRAY Q SIZE SHOULD BE AT LEAST =', I10)
      STOP
   12 CONTINUE
C
C
C     INITIALIZE
      EPS(M)=TOL
      K=M
      WU=0
```

```
      CALL PUTF(M,U1,0)
      CALL PUTF(2*M,F,2)
       ETA=.5
       DELTA=.15
C
C      START OF MAIN LOOP IN WHICH ONE GAUSS-SEIDEL PROJECTED
C      SWEEP ON GRID K IS MADE.
C
    5 ERR=1.E30
    3 ERRP=ERR
      CALL RELSW(K,K+M,ERR)
       IF(WU .LE. 0) ERRBEG=ERR
      WU=WU+4.**(K-M)
      WRITE(6,4)K,ERR,WU
    4 FORMAT(' LEVEL',I2,'   RESIDUAL NORM=',  D10.3,'   WORK=',  F7.3)
      IF(ERR.LT.EPS(K))GOTO 2
      IF (WU.GE.WMAX)RETURN
      IF(K.EQ.1.OR.ERR/ERRP.LT. ETA)GO TO 3
C
C      GO TO COARSER GRID
       IF( K.NE.M .OR. WU.LE.3 ) GOTO 92
       FMU=0.0
       IF( ERR.GT.0 ) FMU=(ERR/ERRBEG)**(1.D0/(WU-1))
       PRINT 91,FMU
91     FORMAT(' ', 20('*'),'END OF CYCLE',20('*'),'MU = ',F8.4)
92     CONTINUE
      CALL RESSW(K,K+M,K+M-1)
      EPS(K-1)=DELTA*ERR
      K=K-1
      CALL PUTSW(K+1,K)
      CALL CORSRE(K,K+M)
      GOTO 5
C
C      GO TO FINER GRID
    2 IF (K.EQ.M)RETURN
      CALL SUBSW(K+1,K)
      CALL INTSW(K,K+1)
      K=K+1
      GOTO 5
      END
C
C
      SUBROUTINE CORSRE(K,KRHS)
C      APPLIES THE DIFFERENCE OPERATOR ON GRID K
C      TO THE GRID FUNCTION IN ARRAY K, AND ADDS THE RESULT TO THE
C      VALUES IN ARRAY KRHS.
C       KRHS    KRHS      K  K,0
C      B     = R       + A  U
C
C      THE RESULT IS STORED IN ARRAY KRHS.
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(200),IRHS(200)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      DO 1 I=2,I1
```

```
      IR=IRHS(I)
      IO=IST(I)
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
      A=-Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
    1 Q(IR+J)=-A-4.*Q(IO+J)
      RETURN
      END
C
C
      SUBROUTINE GRDFN(N,IMAX,JMAX,HH)
C      SETS UP ARRAY N.
C      IMAX     THE DIMENSION IN THE X DIRECTION
C      JMAX     THE DIMENSION IN THE Y DIRECTION
C      HH       THE GRID SIZE
C      THE ARRAY NST CONTAINS THE STARTING ADDRESSES OF THE ARRAYS.
C      THE ARRAY IMX CONTAINS THE MAXIMUM ROW NUMBERS
C      THE ARRAY JMX CONTAINS THE MAXIMUM COL NUMBERS
C      THE ARRAY H   CONTAINS THE GRID SIZES.
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
       COMMON /QDAT/NQSIZE,NQERR
      DATA IQ/1/
      NST(N)=IQ
      IMX(N)=IMAX
      JMX(N)=JMAX
      H(N)=HH
      IQ=IQ+IMAX*JMAX
       IF(IQ.LE.NQSIZE+1) RETURN
       NQERR=IQ-1
      END
C
C
      SUBROUTINE INTSW(KC,KF)
C       INTERPOLATES CORRECTION ON COARSE GRID KC
C      AND ADDS TO SOLUTION ON GRID  KF.
C       KF          KF  KC     KF     KF
C      U   = PHI( I   W   + U    ; U  )
C                  KC
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
       GOTO(1,2,3),NINTSW
C
    1  CALL INTADD(KC,KF)
       RETURN
C
    2  CALL INTADM(KC,KF)
       RETURN
C
    3  CALL INTAPR(KC,KF)
       RETURN
       END
C
C
```

```
      SUBROUTINE INTADD(KC,KF)
C     LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
C     AND ADDS TO SOLUTION ON GRID  KF.
C      KF         KF  KC    KF    KF
C     U   = PHI( I   W   + U   ; U  )
C                 KC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),ISTC(200),ISTF(200)
      CALL KEY(KC,ISTC,IIC,JJC,HC)
      CALL KEY(KF,ISTF,IIF,JJF,HF)
      DO 1 IC=2,IIC
      IF=2*IC-1
      JF=1
      IFO=ISTF(IF)
      IFM=ISTF(IF-1)
      ICO=ISTC(IC)
      ICM=ISTC(IC-1)
      DO 1 JC=2,JJC
      JF=JF+2
      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
      Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
      Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
      Q(IFO+JF-1)=Q(IFO+JF-1)+A
    1 Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
      RETURN
      END
C
      SUBROUTINE INTADM(KC,KF)
C      MODIFICATION #6.
C      LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
C      AND ADDS TO SOLUTION ON GRID  KF.
C      CORRECTION ONLY ADDED IF SOLUTION U ON FINE GRID IS
C      NOT ZERO. SEE (5.15).
C       KF     KF  KC     KF
C      U    = I   U    + U
C             KC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),ISTC(200),ISTF(200)
      CALL KEY(KC,ISTC,IIC,JJC,HC)
      CALL KEY(KF,ISTF,IIF,JJF,HF)
      DO 1 IC=2,IIC
      IF=2*IC-1
      JF=1
      IFO=ISTF(IF)
      IFM=ISTF(IF-1)
      ICO=ISTC(IC)
      ICM=ISTC(IC-1)
      DO 1 JC=2,JJC
      JF=JF+2
      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
      IF(Q(IFO+JF).NE.0)Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
      IF(Q(IFM+JF).NE.0)Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
      IF(Q(IFO+JF-1).NE.0)Q(IFO+JF-1)=Q(IFO+JF-1)+A
```

```
      IF(Q(IFM+JF-1).NE.0)Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
    1  CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE INTAPR(KC,KF)
C        MODIFICATION #1, PHI=MAX(0,U)
C     LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
C     AND ADDS TO SOLUTION ON GRID  KF.
C      KF          KF  KC    KF    KF
C     U   = PHI( I   W   + U   ; U  )
C                 KC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),ISTC(200),ISTF(200)
      CALL KEY(KC,ISTC,IIC,JJC,HC)
      CALL KEY(KF,ISTF,IIF,JJF,HF)
      DO 1 IC=2,IIC
      IF=2*IC-1
      JF=1
      IFO=ISTF(IF)
      IFM=ISTF(IF-1)
      ICO=ISTC(IC)
      ICM=ISTC(IC-1)
      DO 1 JC=2,JJC
      JF=JF+2
      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
      Q(IFO+JF) =AMAX1(0.0D0, Q(IFO+JF)+Q(ICO+JC) )
      Q(IFM+JF) =AMAX1(0.0D0, Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC)) )
      Q(IFO+JF-1)=AMAX1(0.0D0,Q(IFO+JF-1)+A )
    1 Q(IFM+JF-1) =AMAX1(0.0D0, Q(IFM+JF-1)+.5*(A+AM) )
      RETURN
      END
C
      SUBROUTINE KEY(K,IST,IMAX,JMAX,HH)
C      RECOVERS THE INFORMATION ABOUT ARRAY K SET UP BY
C      THE SUBROUTINE GRDFN.
C      THE VALUE OF THE GRID FUNCTION AT THE POINT (I,J)
C      IS ADDRESSED AS U(IST(J)+I).
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
      DIMENSION IST(1)
      IMAX=IMX(K)
      JMAX=JMX(K)
      IS=NST(K)-JMAX-1
      DO 1 I=1,IMAX
      IS=IS + JMAX
    1 IST(I)=IS
      HH=H(K)
      RETURN
      END
C
C
```

```
      SUBROUTINE PUTF(K,F,NH)
C      INSERTS THE VALUES OF THE FUNCTION F
C      EVALUATED AT THE POINTS OF GRID K
C      AND MULTIPLIED BY GRIDSIZE**NH
C      INTO THE ARRAY K.
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(600)
      CALL KEY (K,IST,II,JJ,H)
      H2=H**NH
      DO 1 I=1,II
      DO 1 J=1,JJ
      X=(I-1)*H
      Y=(J-1)*H
    1 Q(IST(I)+J)=F(X,Y)*H2
      RETURN
      END
C
C
      SUBROUTINE PUTSW(KF,KC)
C      THIS SUBROUTINE TRANSFERS THE SOLUTION ON THE FINE GRID
C      KF INTO THE COARSE GRID KC.
C       KC,0    KC  KF
C      U     = I   U
C                KF
C
       COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
       GOTO(1,2),NPUTSW
    1  CALL PUTU(KF,KC)
       RETURN
    2  CALL PUTUNN(KF,KC)
       RETURN
       END
C
C
      SUBROUTINE PUTU(KF,KC)
C      THIS SUBROUTINE INJECTS THE SOLUTION ON THE FINE GRID
C      KF INTO THE COARSE GRID KC.
C       KC,0    KC  KF
C      U     = I   U
C                KF
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      Q(ICO+JC)=          Q(IFO+JF)
    1 CONTINUE
      RETURN
      END
```

```
      C
      C
            SUBROUTINE PUTUNN(KF,KC)
      C          MODIFICATION #2. TRANSFER 0 IF ANY NEIGHBOR ZERO.
             IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON Q(18000),IUF(200),IUC(200)
            CALL KEY(KF,IUF,IIF,JJF,HF)
            CALL KEY(KC,IUC,IIC,JJC,HC)
            DO 1 IC=1,IIC
            IF=2*IC-1
            IFO=IUF(IF)
            ICO=IUC(IC)
            JF=-1
            DO 1 JC=1,JJC
            JF=JF+2
            QTEMP=          Q(IFO+JF)
             IF (IC.EQ.1 .OR. IC.EQ.IIC) GO TO 1
             IF (JC.EQ.1 .OR. JC.EQ.JJC) GO TO 1
             IFP=IUF(IF+1)
             IFM=IUF(IF-1)
             IF(Q(IFP+JF-1).LE.0) QTEMP=0
             IF(Q(IFP+JF+1).LE.0) QTEMP=0
             IF(Q(IFP+JF).LE.0) QTEMP=0
             IF(Q(IFM+JF-1).LE.0) QTEMP=0
             IF(Q(IFM+JF+1).LE.0) QTEMP=0
             IF(Q(IFM+JF).LE.0) QTEMP=0
             IF(Q(IFO+JF-1).LE.0) QTEMP=0
             IF(Q(IFO+JF+1).LE.0) QTEMP=0
      1      Q(ICO+JC)=QTEMP
            RETURN
            END
      C
      C
      C
             SUBROUTINE RELSW(K,KRHS,ERR)
      C      CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
      C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
      C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
             IMPLICIT DOUBLE PRECISION (A-H,O-Z)
             COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
             GOTO (1,2),NRELSW
      C
          1 CALL RELAX(K,KRHS,ERR)
             RETURN
      C
          2 CALL RELXFR(K,KRHS,ERR)
             RETURN
             END
      C
      C
            SUBROUTINE    RELAX(K,KRHS,ERR)
      C      NORMAL RELAXATION
      C      CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
      C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
      C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
      C
             IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      COMMON Q(18000),IST(200),IRHS(200)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      ERR=0.
      DO 1 I=2,I1
      IR=IRHS(I)
      IO=IST(I)
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
       QT=-.25*A
       QN=MAX(0.0,QT)
      ERR=ERR+(QN-Q(IO+J))**2
    1 Q(IO+J)=QN
      ERR=SQRT(ERR)/H
      RETURN
      END
C
      SUBROUTINE    RELXFR(K,KRHS,ERR)
C      "FROZEN" RELAXATION: MODIFICATION # 3
C      CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
C      SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
C      RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
C      DOES NOT CHANGE VALUE OF U ON GRID K
C      IF K<M AND U=0 ON GRID M
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(200),IRHS(200)
       DIMENSION ISTM(100)
C      ASSUMES THAT U AND RHS ARE STORED ON GRIDS SEPARATED BY M
       M=KRHS-K
      CALL KEY(K,IST,II,JJ,H)
       CALL KEY(M,ISTM,IIM,JJM,HM)
       INTERV=2**(M-K)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      ERR=0.
      DO 1 I=2,I1
      IR=IRHS(I)
      IO=IST(I)
       IZM=ISTM(1+INTERV*(I-1))
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
       IF(K.EQ.M) GO TO 10
       QM=Q(IZM+1+INTERV*(J-1))
       IF(QM.EQ.0) GO TO 1
   10  CONTINUE
      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
       QT=-.25*A
       QN=MAX(0.0,QT)
      ERR=ERR+(QN-Q(IO+J))**2
      Q(IO+J)=QN
    1  CONTINUE
```

```
      ERR=SQRT(ERR)/H
      RETURN
      END
C
      SUBROUTINE RESSW(KF,KRF,KRC)
C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
C      IN ARRAY KRF , AND TRANSFERS INTO ARRAY KRC.
C      BEFORE TRANSFER, THE RESIDUAL IS SCALED
C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
C      GRIDSIZE ON GRID KC.
C       KRC     KC    KRF   KF  KF
C      R   = 4*S   ( B   - A   U   )
C               KF
C
      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
      GOTO (1,2,3),NRESSW
C
    1 CALL RESCAL(KF,KRF,KRC)
      RETURN
C
    2 CALL RESCL1(KF,KRF,KRC)
      RETURN
C
    3 CALL RESCAV(KF,KRF,KRC)
      RETURN
      END
C
C
     SUBROUTINE RESCAL(KF,KRF,KRC)
C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
C      BEFORE INJECTION, THE RESIDUAL IS SCALED
C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
C      GRIDSIZE ON GRID KC.
C       KRC     KC    KRF   KF  KF
C      R   = 4*S   ( B   - A   U   )
C               KF
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     COMMON Q(18000),IUF(200),IRF(200),IRC(200)
     CALL KEY(KF,IUF,IIF,JJF,HF)
     CALL KEY(KRF,IRF,IIF,JJF,HF)
     CALL KEY(KRC,IRC,IIC,JJC,HC)
     IIC1=IIC-1
     JJC1=JJC-1
     DO 1 IC=2,IIC1
     ICR=IRC(IC)
     IF=2*IC-1
     JF=1
     IFR=IRF(IF)
     IFO=IUF(IF)
     IFM=IUF(IF-1)
     IFP=IUF(IF+1)
     DO 1 JC=2,JJC1
     JF=JF+2
```

```
      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
    1 Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
      RETURN
      END
C
C
      SUBROUTINE RESCL1(KF,KRF,KRC)
C      MODIFICATION #5  UPDATED JUNE 23 1980
C      USES WEIGHTED RESIDUALS NEAR THE BOUNDARY
C      CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
C      IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
C      BEFORE INJECTION, THE RESIDUAL IS SCALED
C      BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
C      FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
C      GRIDSIZE ON GRID KC.
C       KRC     KC    KRF   KF  KF
C      R   = 4*I   ( B   - A   U   )
C                KF
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       COMMON Q(18000),IUF(200),IRF(200),IRC(200)
       DIMENSION R(9)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KRF,IRF,IIF,JJF,HF)
      CALL KEY(KRC,IRC,IIC,JJC,HC)
      IIC1=IIC-1
      JJC1=JJC-1
      DO 1 IC=2,IIC1
      ICR=IRC(IC)
      IF=2*IC-1
      JF=1
      IFR=IRF(IF)
      IFO=IUF(IF)
      IFM=IUF(IF-1)
      IFP=IUF(IF+1)
      DO 1 JC=2,JJC1
      JF=JF+2
       IF(Q(IFO+JF).EQ.0)GOTO 2
       IF(Q(IFP+JF+1).GT.0 .AND. Q(IFP+JF-1).GT.0 .AND.
     *    Q(IFO+JF+1).GT.0 .AND. Q(IFO+JF-1).GT.0 .AND.
     *    Q(IFM+JF+1).GT.0 .AND. Q(IFM+JF-1).GT.0 .AND.
     *    Q(IFM+JF  ).GT.0 .AND. Q(IFP+JF  ).GT.0 )GOTO 2
       N=0
       DO 3 I1=1,3
       I=IF+I1-2
       DO 3 J1=1,3
       J=JF+J1-2
       N=N+1
      IR=IRF(I)
      IO=IUF(I)
      IM=IUF(I-1)
      IP=IUF(I+1)
      S=Q(IO+J+1)+Q(IO+J-1)+Q(IM+J)+Q(IP+J)
       S=Q(IR+J)+4*Q(IO+J)-S
       IF(Q(IO+J).EQ.0)S=0
       R(N)=S
    3  CONTINUE
```

```
      Q(ICR+JC)=R(5)+.5*(R(2)+R(4)+R(6)+R(8)+
     1    .5*(R(1)+R(3)+R(7)+R(9) ))
      GOTO 1
    2 S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
      Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
    1  CONTINUE
      RETURN
      END
C
C
      SUBROUTINE RESCAV(KF,KRF,KRC)
C      MODIFICATION #4
C      AVERAGES RESIDUALS OVER NEIGHBOURING POINTS
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KRF,IRF,IIF,JJF,HF)
      CALL KEY(KRC,IRC,IIC,JJC,HC)
C      CLEAR COARSE GRID
       DO 9 I=1,IIC
       ICR=IRC(I)
       DO 9 J=1,JJC
    9  Q(ICR+J)=0.
C
      IIF1=IIF-1
      JJF1=JJF-1
      DO 100 IF=2,IIF1
       IC=(IF+1)/2
       IL=IF+1-2*IC
      ICR=IRC(IC)
      IFR=IRF(IF)
      IFO=IUF(IF)
      IFM=IUF(IF-1)
      IFP=IUF(IF+1)
       DO 100 JF=2,JJF1
      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
      RES=(Q(IFR+JF)-S+4.*Q(IFO+JF))
       JC=(JF+1)/2
       JL=JF+1-2*JC
       K=2*IL+JL+1
       GO TO (1,2,3,4),K
    1  Q(ICR+JC)=Q(ICR+JC)+RES
       GO TO 100
    2  RES=RES/2
       Q(ICR+JC)=Q(ICR+JC)+RES
       Q(ICR+JC+1)=Q(ICR+JC+1)+RES
       GO TO 100
    3  RES=RES/2
       Q(ICR+JC)=Q(ICR+JC)+RES
       ICR1=IRC(IC+1)
       Q(ICR1+JC)=Q(ICR1+JC)+RES
       GO TO 100
    4  RES=RES/4
       Q(ICR+JC)=Q(ICR+JC)+RES
       Q(ICR+JC+1)=Q(ICR+JC+1)+RES
       ICR1=IRC(IC+1)
       Q(ICR1+JC)=Q(ICR1+JC)+RES
```

```
      Q(ICR1+JC+1)=Q(ICR1+JC+1)+RES
      GO TO 100
  100 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE SOLPRT(M,MPRINT)
C     PRINTS THE ARRAY M ON THE SUBARRAY MPRINT.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(600)
      DIMENSION QTEM(100)
      CALL KEY (M,IST,II,JJ,H)
      INTERV=2**(M-MPRINT)
      DO 20 J=JJ,1,-INTERV
      L=0
      DO 10 I=1,II,INTERV
C     X AND Y ARE NOT PRINTED HERE, BUT ARE COMPUTED IN
C     CASE A LATER VERSION NEEDS THEM.
      X=(I-1)*H
      Y=(J-1)*H
      L=L+1
      QTEM(L)=Q(IST(I)+J)
   10    CONTINUE
      PRINT *,(QTEM(LL),LL=1,L)
   20    CONTINUE
      RETURN
      END
C
C
      SUBROUTINE SUBSW(KF,KC)
C     THIS SUBROUTINE COMPUTES THE VALUE TRANSFERRED FROM GRID KF TO
C     GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
C      KC    KC    KC  KF
C     W   = U   - I   U
C                  KF
C
      COMMON /SWDAT/NFGSW,NINTSW,NPUTSW,NRELSW,NRESSW
      GOTO(1,2),NPUTSW
    1 CALL SUBTRC(KF,KC)
      RETURN
    2 CALL SUBTNN(KF,KC)
      RETURN
      END
C
C
      SUBROUTINE SUBTRC(KF,KC)
C     THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
C     GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
C      KC    KC    KC  KF
C     W   = U   - I   U
C                  KF
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
```

```
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      Q(ICO+JC)=Q(ICO+JC)-Q(IFO+JF)
    1 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE SUBTNN(KF,KC)
C      MODIFICATION #2. TRANSFER 0 IF ANY NEIGHBOR ZERO.
C      THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
C      GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
C       KC    KC    KC  KF
C      W   = U   - I   U
C                   KF
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      QTEMP=          Q(IFO+JF)
       IF (IC.EQ.1 .OR. IC.EQ.IIC) GO TO 1
       IF (JC.EQ.1 .OR. JC.EQ.JJC) GO TO 1
       IFP=IUF(IF+1)
       IFM=IUF(IF-1)
       IF(Q(IFP+JF-1).LE.0) QTEMP=0
       IF(Q(IFP+JF+1).LE.0) QTEMP=0
       IF(Q(IFP+JF).LE.0) QTEMP=0
       IF(Q(IFM+JF-1).LE.0) QTEMP=0
       IF(Q(IFM+JF+1).LE.0) QTEMP=0
       IF(Q(IFM+JF).LE.0) QTEMP=0
       IF(Q(IFO+JF-1).LE.0) QTEMP=0
       IF(Q(IFO+JF+1).LE.0) QTEMP=0
    1  Q(ICO+JC)=Q(ICO+JC)-QTEMP
      RETURN
      END
C
C
```

```
                =====  APPX-C-PFMG  =====

 1.     C      ********************************************************************
 2.     C
 3.     C      THIS PROGRAM SOLVES THE PROBLEM OF POROUS FLOW THROUGH A
 4.     C      RECTANGULAR DAM OF HEIGHT Y1 AND WIDTH A.
 5.     C      THE RESERVOIR TO THE RIGHT OF THE DAM IS OF HEIGHT Y2.
 6.     C
 7.     C      WRITTEN BY ACHI BRANDT AND COLIN CRYER AUGUST 1980
 8.     C
 9.     C      THIS PROGRAM WAS USED TO COMPUTE THE RESULTS IN
10.     C      SECTION 6 OF THE MRC REPORT.
11.     C
12.     C      ADDITIONAL PARAMETERS USED ARE:
13.     C      NX0      THE NUMBER OF GRID INTERVALS IN THE X-DIRECTION IN
14.     C               THE COARSEST GRID, GRID 1.
15.     C      NY0      THE NUMBER OF GRID INTERVALS IN THE Y-DIRECTION IN
16.     C               THE COARSEST GRID, GRID 1.
17.     C      H0       THE GRID SIZE IN THE COARSEST GRID, GRID 1.
18.     C      M        THE NUMBER OF GRIDS TO BE USED.
19.     C      LIN      THE STARTING GRID. LIN.GE.2
20.     C      TOL      THE TOLERANCE
21.     C      RATIO    TOLERANCE ON GRID L IS TOLL=TOL*RATIO**L
22.     C      WMAXM    THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
23.     C               FINEST GRID. COMPUTATION TERMINATES WHEN WMAXM IS
24.     C               EXCEEDED.
25.     C      WMAX     THE MAXIMUM NUMBER OF WORK UNITS PERMITTED ON THE
26.     C               GRID L<M. COMPUTATION ON GRID L TERMINATES WHEN WMAX IS
27.     C               EXCEEDED.
28.     C      MPRINT   THE GRID TO BE PRINTED AT THE END OF THE COMPUTATION.
29.     C               THAT IS, WE PRINT THE MPRINT SUBSET OF THE FINAL ANSWER
30.     C               ON THE GRID M.
31.     C      NQSIZE   SIZE OF ARRAY Q
32.     C               MUST BE CHANGED FOR LARGE PROBLEMS BY EDITING PROGRAM
33.     C               =18000 FOR DAM PROBLEM M=2,3,4,5,6
34.     C      NR1      AFTER NR1 RELAXATIONS ON THE GRID K+1 THERE IS A
35.     C               TRANSFER TO GRID K.
36.     C      NR2      AFTER A TOTAL NUMBER OF NR2 RELAXATIONS ON GRID K
37.     C               THERE IS A TRANSFER TO GRID K+1
38.     C      NCYC     MAXIMUM NUMBER OF CYCLES ON LEVEL L, LIN< L<M
39.     C      NCYCLN   MAXIMUM NUMBER OF CYCLES ON LEVEL LIN
40.     C      NCYCM    MAXIMUM NUMBER OF CYCLES ON LEVEL M
41.     C      ETA      IF ERR.GE.ETA*ERRP GO TO COARSER GRID
42.     C      DELTA    EPS(K-1)=DELTA*(ERROR ERR ON GRID K)
43.     C      PREC     EPS(L)=MAX(PREC*TAU(L-1),TOL*RATIO**L)
44.     C      PRECM    EPS(M)=MAX(PRECM*TAU(M-1),TOL*RATIO**M)
45.     C
46.     C      WE CAN ALSO DO TAU EXTRAPOLATION:
47.     C      ITAU     IF ITAU=1 DO TAU EXTRAPOLATION
48.     C      PT       ORDER OF EXTRAPOLATION
49.     C
50.     C      SWITCHES
51.     C
52.     C      NFGSW    USED IN SUBRUTINES F,G,SOLRED
53.     C      NFGSW    =1 DAM PROBLEM
54.     C               =2 PROBLEM (5.3),(5.4).
55.     C
56.     C
57.     C      NINTSW   =1 INJECTION. SUBROUTINE INTADD
```

```
C              =2 MODIFICATION #6. SUBROUTINE INTADM
C                 CORRECTION ONLY ADDED WHEN U.NE.0. SEE (5.15).
C
C      NRESSW  =1 INJECTION. SUBROUTINE RESCAL
C              =2 MODIFICATION #5. SUBROUTINE RESCL1
C                 USES WEIGHTED RESIDUALS NEAR BOUNDARY.
C                 RESIDUALS WITH U<0 SET EQUAL TO ZERO
C
C
C      ALL THE PARAMETERS ARE SET IN THE PROGRAM, BUT THEIR VALUES
C      CAN BE RESET ON THE NAMELIST INPUT CARD WHICH IS READ IN
C      BY THE PROGRAM.
C      THE NAMELIST CARD MUST BE PROVIDED AS INPUT.
C
C      THE PROGRAM SETS UP STORAGE FOR THE SOLUTIONS AND RIGHT
C      HAND SIDES.
C      THE SOLUTIONS ARE STORED IN ARRAYS 1 TO M.
C      THE RIGHT HAND SIDES ( OR, SOMETIMES THE RESIDUALS )
C      ARE STORED IN ARRAYS M+1 TO 2*M.
C
C      THE EXACT SOLUTION (WHEN KNOWN) IS STORED IN GRID NGRSOL
C      THE VALUES OF TAU ARE STORED IN GRIDS 2M+1 TO 3M-1
C
C      ************************************************************
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       EXTERNAL G,F
       COMMON /PRBDAT/Y1,Y2,A,R
       COMMON /QDAT/NQSIZE,NQERR
       COMMON /SOLTAU/M,NGRSOL,PT
       COMMON /SWDAT/NFGSW,NINTSW,NRESSW
       NAMELIST /INDAT/NX0,NY0,H0,M,LIN,NR1,NR2,ETA,DELTA
     *  ,TOL,RATIO,PREC,PRECM,WMAX,WMAXM,NCYC,NCYCLN,NCYCM,ITAU,PT,
     *  MPRINT,Y1,Y2,A,R
     * ,NFGSW,NINTSW,NRELSW,NRESSW
       CHARACTER ITITLE(80)
C
C      READ IN AND PRINT TITLE CARDS
C      FINISH READING TITLE WHEN LAST CARD IS BLANK
C      FINISH RUN WHEN TITLE CARD IS BLANK
       PRINT 18
   18  FORMAT(1H1)
       NC=0
    5  READ 10,(ITITLE(I),I=1,80)
   10  FORMAT(80A1)
       NC=NC+1
       PRINT 11,(ITITLE(I),I=1,80)
   11  FORMAT(1H ,80A1)
       DO 12 I=1,80
       IF (ITITLE(I).NE.' ')GOTO 5
   12  CONTINUE
       IF(NC.EQ.1) STOP
C
       NQSIZE=18000
       NFGSW=1
      NINTSW=1
       NRESSW=1
       Y1=24
```

```
      Y2=4
      A=16
      R=32.D0/15.D0
      NX0=2
      NY0=3
      H0=8.
      M=6
      LIN=2
      NR1=2
      NR2=3
      ETA=10.
      DELTA=0
      TOL=0
      RATIO=1
      PREC=0
      PRECM=1
      WMAX=30.
      WMAXM=40
      NCYC=1
      NCYCLN=3
      NCYCM=10
      ITAU=0
      PT  =2
      MPRINT=2
      READ(5,INDAT)
      WRITE(6,INDAT)
C     PRINT MODIFICATION NUMBERS
      PRINT 100
100   FORMAT( '0  *** THE FOLLOWING MODIFICATIONS WERE USED *** '/)
      IF(NINTSW.EQ.2) PRINT 106
      IF(NINTSW.EQ.3) PRINT 101
      IF(NRELSW.EQ.2) PRINT 103
      IF(NRESSW.EQ.2) PRINT 105
      IF(NRESSW.EQ.3) PRINT 104
101   FORMAT('0', 'MODIFICATION NUMBER 1')
103   FORMAT('0', 'MODIFICATION NUMBER 3')
104   FORMAT('0', 'MODIFICATION NUMBER 4')
105   FORMAT('0', 'MODIFICATION NUMBER 5')
106   FORMAT('0', 'MODIFICATION NUMBER 6')
      PRINT 110
110   FORMAT( ' ************* ')
C     SET TIME TO ZERO
      CALL URTIMS(0.0)
      CALL PFMG(NX0,NY0,H0,LIN,NR1,NR2,ETA,DELTA
     *  ,TOL,RATIO,PREC,PRECM,WMAX,WMAXM,NCYC,NCYCLN,NCYCM,ITAU,
     *  MPRINT,G,F)
      T=URTIMG('ELAPSE')
   19 FORMAT(1H0, ' GRID-M SOLUTION',//)
      PRINT 19
      CALL SOLPRT(M,MPRINT)
      PRINT 20
   20 FORMAT(1H1, ' GRID-7 SOLUTION',//)
      CALL SOLPRT(NGRSOL,MPRINT)
      STOP
      END
C
C
```

```
      DOUBLE PRECISION FUNCTION F(X,Y)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /PRBDAT/Y1,Y2,A,R
      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
C     THIS SUBROUTINE COMPUTES THE RIGHT HAND SIDE OF THE
C     GOVERNING POISSON EQUATION DEL*DEL U=F.
      GOTO( 1,2),NFGSW
C
C     DAM PROBLEM
    1 CONTINUE
      F=1.
      RETURN
C
C     PROBLEM OF SECTION 5: (5.3) AND (5.4)
    2 CONTINUE
      D=2.5*R
      A=DMAX1(0.D0,D-R*X-Y)
      B=X+Y
      C=2*(R**2+1)
      F=(C-2.*A*A)*DCOS(B) +4*(R+1)*A*DSIN(B)+2*C
      RETURN
      END
C
C
      DOUBLE PRECISION FUNCTION G(X,Y)
C     THIS SUBROUTINE COMPUTES THE BOUNDARY DATA AND THE
C     INITIAL APPROXIMATION TO THE SOLUTION U.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /PRBDAT/Y1,Y2,A,R
      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
      GOTO( 1,2),NFGSW
C
C     DAM PROBLEM
C     THE INITIAL APPROXIMATION IS OBTAINED BY LINEAR INTERPOLATION
C     IN THE X-DIRECTION BETWEEN THE GIVEN BOUNDARY DATA.
    1 CONTINUE
      G1=.5*(Y1-Y)**2
      G2=.5*(Y2-Y)**2
      IF( Y.GE.Y2) G2=0
      G=(G1*(A-X)+ G2*X)/A
      RETURN
C
C     PROBLEM OF SECTION 5: (5.3) AND (5.4)
C     INITIAL APPROXIMATION IS A PERTURBATION OF EXACT SOLUTION
    2 CONTINUE
      D=2.5*R
      A=DMAX1(0.D0,D-R*X-Y)
      B=X+Y
      G=A*A*(DCOS(B)+2)
      G=G+X*(3-X)*Y*(2-Y)*10
      RETURN
       END
C
       SUBROUTINE PFMG(NX0,NY0,H0,LIN,NR1,NR2,ETA,DELTA
     *  ,TOL,RATIO,PREC,PRECM,WMAX,WMAXM,NCYC,NCYCLN,NCYCM,ITAU,
     *  MPRINT,U1,F)
C     THIS SUBROUTINE IS THE MAIN FULL MULTIGRID SUBROUTINE.
```

```
C       IT INITIALIZES THE PROBLEM, AND REPEATEDLY CALLS
C       THE SUBROUTINES RELAX,RESCAL,PUTU,CORSRE,SUBTRC,AND INTADD.
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /QDAT/NQSIZE,NQERR
        EXTERNAL U1,F
        DIMENSION EPS(10),IR2(10)
        COMMON /SOLTAU/M,NGRSOL,PT
C
C
C       SET UP ARRAYS 1 TO M FOR THE SOLUTIONS
C       AND ARRAYS M+1 TO 2*M FOR THE RIGHT HAND SIDES,
C       AND ARRAYS 2M+1 TO 3M-1 FOR TAU ARRAYS AND
C       SET ASIDE SPACE FOR GRID-7 SOLUTION IN 3M=NGRSOL GRID
C       AND CHECK THAT Q ARRAY IS LARGE ENOUGH
        NQERR=0
        DO 1 K=1,M
        K2=2**(K-1)
        CALL GRDFN(K,NX0*K2+1,NY0*K2+1,H0/K2)
        CALL GRDFN(K+M,NX0*K2+1,NY0*K2+1,H0/K2)
     1  CALL GRDFN(K+2*M,NX0*K2+1,NY0*K2+1,H0/K2)
        NGRSOL=3*M
        PRINT 90,NQSIZE
   90     FORMAT(' SIZE OF Q ARRAY = ', I10)
        IF(NQERR.EQ.0)GOTO 92
         PRINT 91,NQERR
    91     FORMAT(' *** ERROR IN GRDFN *** ARRAY Q NOT LARGE ENOUGH ***',
     * /,' ARRAY Q SIZE SHOULD BE AT LEAST =', I10)
        STOP
   92   CONTINUE
C
C
        CALL SOLRED
C
C       INITIALIZE
        WU=0
        CALL PUTF(LIN,U1,0)
        DO 10 L=LIN,M
C
C       BEGIN NEW FINEST LEVEL
C
        PRINT 6,L
     6  FORMAT(1H0,60(1H.),I3,2X,60(1H.)/)
        CALL PUTF(L+M,F,2)
        TOLL=TOL*(RATIO**L)
        EPS(L)=TOLL
        WU=.25*WU
        NCYCL=NCYC
        IF(L.EQ.M)NCYCL=NCYCM
        ICYC=0
        WMAXL=WMAX
        IF(L.EQ.M)WMAXL=WMAXM
        PRECL=PREC
        IF(L.EQ.M)PRECL=PRECM
C
C
        K=L
        IR2(L)=0
```

```
286.      C
287.      C       BEGIN A NEW WORK LEVEL
288.      C
289.          5   IR1=0
290.              ERR=1.E30
291.      C
292.      C       RELAX ONCE ON GRID K
293.          3   ERRP=ERR
294.              CALL RELAX(K,K+M,ERR)
295.              WU=WU+4.**(K-L)
296.              IR1=IR1+1
297.              IR2(K)=IR2(K)+1
298.              WRITE(6,40)K,ERR,WU,IR1,IR2(K)
299.         40   FORMAT(' LEVEL',I2,'   RESIDUAL NORM=',  D10.3,'   WORK=',  F7.3
300.           * ,' IR1= ',I2,' IR2(K)=',I2)
301.      C
302.      C       DECIDE WHICH GRID TO USE NEXT
303.              IF (WU.GE.WMAXL)GOTO 20
304.              IF(ERR.LT.EPS(K))GOTO 2
305.              IF(IR2(K).NE.NR2)GOTO 8
306.              IF( K.LT.L)GOTO 2
307.      C
308.              ICYC=ICYC+1
309.              IF(ICYC.EQ.NCYCL .AND. L.NE.LIN)GOTO 20
310.              IF(ICYC.EQ.NCYCLN .AND. L.EQ.LIN)GOTO 20
311.              IR2(L)=0
312.              IR1=0
313.      C
314.          8   IF(IR1.EQ.NR1)GOTO 4
315.               IF(IR1.EQ.1.OR.ERR.LT. ERRP*ETA)GO TO 3
316.      C
317.      C       GO TO COARSER GRID
318.          4   IF(K.EQ.1)GOTO 3
319.              CALL RESSW(K,K+M,K+M-1)
320.              CALL RESBW(K,K+M,K+2*M-1)
321.              EPS(K-1)=DELTA*ERR
322.              K=K-1
323.              CALL PUTU(K+1,K)
324.              CALL CORSRE(K,K+M)
325.              ITAUEX=0
326.              IF((ITAU.EQ.1).AND.(L.GT.LIN).AND.(K.EQ.L-1))ITAUEX=1
327.              CALL TAUCAM(K,K+M,K+2*M,ITAUEX,TAUGNM)
328.              PRINT 60,TAUGNM,K
329.         60   FORMAT(50X,'GREEN NORM OF TAU-Z =',E12.3,5X,'K=',I2)
330.              IF(K.EQ.(L-1))EPS(L)=DMAX1(PRECL*TAUGNM,TOLL)
331.              IR2(K)=0
332.              GOTO 5
333.      C
334.      C       GO TO FINER GRID
335.          2   IF(K.EQ.L)GOTO 20
336.              CALL SUBTRC(K+1,K)
337.              CALL INTSW(K,K+1)
338.              K=K+1
339.              GOTO 5
340.      C
341.      C
342.      C       FINISHED WITH LEVEL L
```

```
      20    CONTINUE
C
C     THE NEXT SEVEN STATEMENTS COMPUTE THE GREEN NORM OF TAU
C      AND THE GREEN AND L-INFINITY NORMS OF THE ERROR
C        ( IF ACCURATE SOLUTION IS KNOWN )
      11  CALL RESSW(L,L+M,L+M-1)
          CALL RESBW(L,L+M,L+2*M-1)
          CALL PUTU(L,L-1)
          CALL CORSRE(L-1,L-1+M)
          CALL TAUCAM(L-1,L-1+M,L-1+2*M,0,TAUGNM)
          K=L-1
          PRINT 60,TAUGNM,K
          CALL DIFFMX(L)
C
C
          IF(L.EQ.M)GOTO 10
          CALL INTRP3(L,L+1)
          CALL PUTB(U1,L+1)
C
      10  CONTINUE
          RETURN
          END
C
C
          SUBROUTINE CORSRE(K,KRHS)
C         APPLIES THE DIFFERENCE OPERATOR ON GRID K
C         TO THE GRID FUNCTION IN ARRAY K, AND ADDS THE RESULT TO THE
C         VALUES IN ARRAY KRHS.
C          KRHS    KRHS      K  K,0
C         B     = R       + A  U
C
C         THE RESULT IS STORED IN ARRAY KRHS.
          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
          COMMON Q(18000),IST(200),IRHS(200)
          CALL KEY(K,IST,II,JJ,H)
          CALL KEY(KRHS,IRHS,II,JJ,H)
          I1=II-1
          J1=JJ-1
          DO 1 I=2,I1
          IR=IRHS(I)
          IO=IST(I)
          IM=IST(I-1)
          IP=IST(I+1)
          DO 1 J=2,J1
          A=-Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
        1 Q(IR+J)=-A-4.*Q(IO+J)
          RETURN
          END
C
C
          SUBROUTINE DIFFMX(K)
C         NOT TIMED
C         COMPARES SOLUTION ON GRID K WITH ACCURATE SOLUTION
C         STORED IN GRID NGRSOL
          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
          COMMON Q(18000),IST(200),ISTA(200)
          COMMON /SOLTAU/M,NGRSOL,PT
```

```
      TIME=URTIMG(0)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(NGRSOL,ISTA,IIA,JJA,HA)
      DIFMX=0.
      DIFGNM=0
      SOLMX=0.
      SOLGNM=0
      INTERV=(IIA-1)/(II-1)
      DO 1 I=1,II
      X=(I-1)*H
      IA=(I-1)*INTERV+1
      DO 1 J=1,JJ
      Y=(J-1)*H
      JA=(J-1)*INTERV+1
      DIF=ABS(Q(ISTA(IA)+JA)  -Q(IST(I)+J))
      DIFGNM=DIFGNM+DIF*DIF
      SOL=ABS(Q(ISTA(IA)+JA))
      SOLGNM=SOLGNM+SOL*SOL
      SOLMX=AMAX1(SOL,SOLMX)
    1 DIFMX=AMAX1(DIF,DIFMX)
      DIFGNM=SQRT(DIFGNM)/H
      PRINT 101,DIFMX,DIFGNM
  101 FORMAT(15X,' SOLUTION ERROR: L INFINITY NORM =',E13.5,
     * 5X,'GNORM =  ',E13.5)
      SOLGNM=SQRT(SOLGNM)/H
      PRINT 102,SOLMX,SOLGNM
  102 FORMAT(15X,' SOLUTION      : L INFINITY NORM =',E13.5,
     * 5X,'GNORM =  ',E13.5)
      PRINT 103,DIFMX/SOLMX,DIFGNM/SOLGNM
  103 FORMAT(15X,' RELATIVE ERROR: L INFINITY NORM =',E13.5,
     * 5X,'GNORM =  ',E13.5)
      CALL URTIMS(TIME)
      RETURN
      END
C
C
      SUBROUTINE GRDFN(N,IMAX,JMAX,HH)
C     SETS UP ARRAY N.
C     IMAX    THE DIMENSION IN THE X DIRECTION
C     JMAX    THE DIMENSION IN THE Y DIRECTION
C     HH      THE GRID SIZE
C     THE ARRAY NST CONTAINS THE STARTING ADDRESSES OF THE ARRAYS.
C     THE ARRAY IMX CONTAINS THE MAXIMUM ROW NUMBERS
C     THE ARRAY JMX CONTAINS THE MAXIMUM COL NUMBERS
C     THE ARRAY H   CONTAINS THE GRID SIZES.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
      COMMON /QDAT/NQSIZE,NQERR
      DATA IQ/1/
      NST(N)=IQ
      IMX(N)=IMAX
      JMX(N)=JMAX
      H(N)=HH
      IQ=IQ+IMAX*JMAX
      IF(IQ.LE.NQSIZE+1) RETURN
      NQERR=IQ-1
```

```
      END
C
C
      SUBROUTINE INTSW(KC,KF)
C      INTERPOLATES CORRECTION ON COARSE GRID KC
C     AND ADDS TO SOLUTION ON GRID  KF.
C      KF          KF  KC     KF     KF
C     U   = PHI( I   W   + U    ; U  )
C                 KC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
      GOTO(1,2),NINTSW
C
    1 CALL INTADD(KC,KF)
      RETURN
C
    2 CALL INTADM(KC,KF)
      RETURN
      END
C
C
      SUBROUTINE INTADD(KC,KF)
C     LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
C     AND ADDS TO SOLUTION ON GRID  KF.
C      KF          KF  KC     KF     KF
C     U   = PHI( I   W   + U    ; U  )
C                 KC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),ISTC(200),ISTF(200)
      CALL KEY(KC,ISTC,IIC,JJC,HC)
      CALL KEY(KF,ISTF,IIF,JJF,HF)
      DO 1 IC=2,IIC
      IF=2*IC-1
      JF=1
      IFO=ISTF(IF)
      IFM=ISTF(IF-1)
      ICO=ISTC(IC)
      ICM=ISTC(IC-1)
      DO 1 JC=2,JJC
      JF=JF+2
      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
      Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
      Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
      Q(IFO+JF-1)=Q(IFO+JF-1)+A
    1 Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
      RETURN
      END
C
      SUBROUTINE INTADM(KC,KF)
C     MODIFICATION #6.
C     LINEARLY INTERPOLATES CORRECTION ON COARSE GRID KC
C     AND ADDS TO SOLUTION ON GRID  KF.
C     CORRECTION ONLY ADDED IF SOLUTION U ON FINE GRID IS
C     NOT ZERO. SEE (5.15).
```

```
514.      C        KF     KF  KC     KF
515.      C       U   = I   U   + U
516.      C              KC
517.      C
518.              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
519.              COMMON Q(18000),ISTC(200),ISTF(200)
520.              CALL KEY(KC,ISTC,IIC,JJC,HC)
521.              CALL KEY(KF,ISTF,IIF,JJF,HF)
522.              DO 1 IC=2,IIC
523.              IF=2*IC-1
524.              JF=1
525.              IFO=ISTF(IF)
526.              IFM=ISTF(IF-1)
527.              ICO=ISTC(IC)
528.              ICM=ISTC(IC-1)
529.              DO 1 JC=2,JJC
530.              JF=JF+2
531.              A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
532.              AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
533.              IF(Q(IFO+JF).NE.0)Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
534.              IF(Q(IFM+JF).NE.0)Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
535.              IF(Q(IFO+JF-1).NE.0)Q(IFO+JF-1)=Q(IFO+JF-1)+A
536.              IF(Q(IFM+JF-1).NE.0)Q(IFM+JF-1) = Q(IFM+JF-1)+.5*(A+AM)
537.            1 CONTINUE
538.              RETURN
539.              END
540.      C
541.      C
542.      C
543.              SUBROUTINE INTRP3(KC,KF)
544.      C       PERFORMS CUBIC INTERPOLATION
545.              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
546.              COMMON Q(18000),IUF(200),IUC(200)
547.              CALL KEY(KF,IUF,IIF,JJF,HF)
548.              CALI KEY(KC,IUC,IIC,JJC,HC)
549.      C
550.      C        KF     KF  KC     KF
551.      C       U   = J   U   + U
552.      C              KC
553.      C
554.      C       INTERPOLATE IN COARSE COLUMNS USING COARSE COLUMN DATA
555.              DO 20 IC=1,IIC
556.              IF=2*IC-1
557.              IFO=IUF(IF)
558.              ICO=IUC(IC)
559.              Q(IFO+1)=Q(ICO+1)
560.      C       FIRST POINT IN COLUMN. USE EQU (6.3)
561.              Q(IFO+2)=(5*Q(ICO+1)+15*Q(ICO+2)-5*Q(ICO+3)+Q(ICO+4))/16
562.              JJC2=JJC-2
563.              DO 10 JC=2,JJC2
564.              JF=2*JC-1
565.              Q(IFO+JF)=Q(ICO+JC)
566.      C
567.      C       INTERIOR POINT IN COLUMN. USE EQU (6.2)
568.              Q(IFO+JF+1)=(-Q(ICO+JC-1)+9*Q(ICO+JC)
569.             *   +9*Q(ICO+JC+1)-Q(ICO+JC+2))/16.
570.           10 CONTINUE
```

```
      Q(IFO+JJF-2)=Q(ICO+JJC-1)
      Q(IFO+JJF-1)=(Q(ICO+JJC-3)-5*Q(ICO+JJC-2)
C
C     LAST POINT IN COLUMN. USE EQU (6.3)
     *  +15*Q(ICO+JJC-1)+5*Q(ICO+JJC))/16
      Q(IFO+JJF)=Q(ICO+JJC)
   20 CONTINUE
C
C     INTERPOLATE IN INTERMEDIATE FINE COLUMNS
C     USING ROW DATA
C
C     FIRST COLUMN. USE EQU (6.3)
      IM1=IUF(1)
      IO=IUF(2)
      IP1=IUF(3)
      IP3=IUF(5)
      IP5=IUF(7)
      DO 30 J=1,JJF
      Q(IO+J)=(5*Q(IM1+J)+15*Q(IP1+J)
     * -5*Q(IP3+J)+Q(IP5+J))/16.
   30 CONTINUE
C
C     INTERMEDIATE COLUMNS. USE EQU (6.2)
      IIF3=IIF-3
      DO 40 I=4,IIF3,2
      IM3=IUF(I-3)
      IM1=IUF(I-1)
      IO=IUF(I)
      IP1=IUF(I+1)
      IP3=IUF(I+3)
      DO 40 J=1,JJF
      Q(IO+J)=(-Q(IM3+J)+9*Q(IM1+J)
     *  +9*Q(IP1+J)-Q(IP3+J))/16.
   40 CONTINUE
C
C     LAST COLUMN. USE EQU (6.3)
      IM5=IUF(IIF-6)
      IM3=IUF(IIF-4)
      IM1=IUF(IIF-2)
      IO=IUF(IIF-1)
      IP1=IUF(IIF)
      DO 50 J=1,JJF
      Q(IO+J)=(Q(IM5+J)-5*Q(IM3+J)
     *  +15*Q(IM1+J)+5*Q(IP1+J))/16
   50 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE KEY(K,IST,IMAX,JMAX,HH)
C     RECOVERS THE INFORMATION ABOUT ARRAY K SET UP BY
C     THE SUBROUTINE GRDFN.
C     THE VALUE OF THE GRID FUNCTION AT THE POINT (I,J)
C     IS ADDRESSED AS U(IST(J)+I).
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
```

```
      DIMENSION IST(1)
      IMAX=IMX(K)
      JMAX=JMX(K)
      IS=NST(K)-JMAX-1
      DO 1 I=1,IMAX
      IS=IS + JMAX
    1 IST(I)=IS
      HH=H(K)
      RETURN
      END
C
C
      SUBROUTINE PUTB(F,K)
C     INSERTS THE BOUNDARY VALUES OF THE FUNCTION F
C     EVALUATED AT THE POINTS OF GRID K
C     INTO THE ARRAY K.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000)
      DIMENSION IST(200)
      CALL KEY(K,IST,II,JJ,H)
      II1=II-1
      DO 1 J=1,JJ
      X=0.
      Y=(J-1)*H
      Q(IST(1)+J)=F(X,Y)
      X=(II-1)*H
      Q(IST(II)+J)=F(X,Y)
    1 CONTINUE
      DO 2 I=2,II1
      Y=0.
      X=(I-1)*H
      Q(IST(I)+1)=F(X,Y)
      Y=(JJ-1)*H
      Q(IST(I)+JJ)=F(X,Y)
    2 CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE PUTF(K,F,NH)
C     INSERTS THE VALUES OF THE FUNCTION F
C     EVALUATED AT THE POINTS OF GRID K
C     AND MULTIPLIED BY GRIDSIZE**NH
C     INTO THE ARRAY K.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(600)
      CALL KEY (K,IST,II,JJ,H)
      H2=H**NH
      DO 1 I=1,II
      DO 1 J=1,JJ
      X=(I-1)*H
      Y=(J-1)*H
    1 Q(IST(I)+J)=F(X,Y)*H2
      RETURN
```

```
      END
C
C
      SUBROUTINE PUTU(KF,KC)
C     THIS SUBROUTINE INJECTS THE SOLUTION ON THE FINE GRID
C     KF INTO THE COARSE GRID KC.
C      KC,0    KC  KF
C     U     = I   U
C              KF
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      Q(ICO+JC)=          Q(IFO+JF)
    1 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE    RELAX(K,KRHS,ERR)
C     NORMAL RELAXATION
C     CARRIES OUT ONE GAUSS-SEIDEL PROJECTED
C     SWEEP ON THE GRID K WITH RIGHT HAND SIDE IN ARRAY KRHS.
C     RETURNS WITH ERR= G-NORM OF THE DYNAMIC RESIDUALS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IST(200),IRHS(200)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      ERR=0.
      DO 1 I=2,I1
      IR=IRHS(I)
      IO=IST(I)
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
      A=Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
      QT=-.25*A
      QN=MAX(0.0,QT)
      ERR=ERR+(QN-Q(IO+J))**2
    1 Q(IO+J)=QN
      ERR=SQRT(ERR)/H
      RETURN
      END
C
      SUBROUTINE RESBW(KF,KRF,KRC)
C     SAME AS RESSW EXCEPT THAT ONLY THE RHS B IS TREATED
```

```
      C     CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
      C     IN ARRAY KRF , AND TRANSFERS INTO ARRAY KRC.
      C     BEFORE TRANSFER, THE RESIDUAL IS SCALED
      C     BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
      C     FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
      C     GRIDSIZE ON GRID KC.
      C      KRC      KC     KRF
      C     R   = 4*S    ( B          )
      C              KF
      C
            COMMON /SWDAT/NFGSW,NINTSW,NRESSW
            GOTO (1,2),NRESSW
      C
          1 CALL RESBAL(KF,KRF,KRC)
            RETURN
      C
          2 CALL RESBL1(KF,KRF,KRC)
            RETURN
            END
      C
      C
            SUBROUTINE RESBAL(KF,KRF,KRC)
      C     SAME AS RESCAL EXCEPT THAT ONLY RHS B IS TREATED
      C     CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
      C     IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
      C     BEFORE INJECTION, THE RESIDUAL IS SCALED
      C     BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
      C     FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
      C     GRIDSIZE ON GRID KC.
      C      KRC      KC     KRF
      C     R   = 4*S    ( B          )
      C              KF
      C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON Q(18000),IUF(200),IRF(200),IRC(200)
            CALL KEY(KF,IUF,IIF,JJF,HF)
            CALL KEY(KRF,IRF,IIF,JJF,HF)
            CALL KEY(KRC,IRC,IIC,JJC,HC)
            IIC1=IIC-1
            JJC1=JJC-1
            DO 1 IC=2,IIC1
            ICR=IRC(IC)
            IF=2*IC-1
            JF=1
            IFR=IRF(IF)
            IFO=IUF(IF)
            IFM=IUF(IF-1)
            IFP=IUF(IF+1)
            DO 1 JC=2,JJC1
            JF=JF+2
          1 Q(ICR+JC)=4.*(Q(IFR+JF))
            RETURN
            END
      C
      C
            SUBROUTINE RESBL1(KF,KRF,KRC)
      C     SAME AS RESCL1 EXCEPT THAT ONLY RHS B IS TREATED
```

```
      C     MODIFICATION #5 UPDATED JUNE 23 1980
      C     USES WEIGHTED RESIDUALS NEAR THE BOUNDARY
      C     CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
      C     IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
      C     BEFORE INJECTION, THE RESIDUAL IS SCALED
      C     BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
      C     FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
      C     GRIDSIZE ON GRID KC.
      C      KRC     KC    KRF
      C     R   = 4*S   ( B         )
      C              KF
      C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON Q(18000),IUF(200),IRF(200),IRC(200)
            DIMENSION R(9)
            CALL KEY(KF,IUF,IIF,JJF,HF)
            CALL KEY(KRF,IRF,IIF,JJF,HF)
            CALL KEY(KRC,IRC,IIC,JJC,HC)
            IIC1=IIC-1
            JJC1=JJC-1
            DO 1 IC=2,IIC1
            ICR=IRC(IC)
            IF=2*IC-1
            JF=1
            IFR=IRF(IF)
            IFO=IUF(IF)
            IFM=IUF(IF-1)
            IFP=IUF(IF+1)
            DO 1 JC=2,JJC1
            JF=JF+2
            IF(Q(IFO+JF).EQ.0)GOTO 2
            IF(Q(IFP+JF+1).GT.0 .AND. Q(IFP+JF-1).GT.0 .AND.
           *   Q(IFO+JF+1).GT.0 .AND. Q(IFO+JF-1).GT.0 .AND.
           *   Q(IFM+JF+1).GT.0 .AND. Q(IFM+JF-1).GT.0 .AND.
           *   Q(IFM+JF  ).GT.0 .AND. Q(IFP+JF  ).GT.0 )GOTO 2
            N=0
            DO 3 I1=1,3
            I=IF+I1-2
            DO 3 J1=1,3
            J=JF+J1-2
            N=N+1
            IR=IRF(I)
            IO=IUF(I)
            IM=IUF(I-1)
            IP=IUF(I+1)
            S=Q(IR+J)
            IF(Q(IO+J).EQ.0)S=0
            R(N)=S
          3 CONTINUE
            Q(ICR+JC)=R(5)+.5*(R(2)+R(4)+R(6)+R(8)+
           *   .5*(R(1)+R(3)+R(7)+R(9) ))
            GOTO 1
          2 Q(ICR+JC)=4.*Q(IFR+JF)
          1 CONTINUE
            RETURN
            END
      C
```

```
      C
            SUBROUTINE RESSW(KF,KRF,KRC)
      C     CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
      C     IN ARRAY KRF , AND TRANSFERS INTO ARRAY KRC.
      C     BEFORE TRANSFER, THE RESIDUAL IS SCALED
      C     BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
      C     FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
      C     GRIDSIZE ON GRID KC.
      C      KRC     KC    KRF   KF  KF
      C     R   = 4*S   ( B   - A   U   )
      C               KF
      C
            COMMON /SWDAT/NFGSW,NINTSW,NRESSW
            GOTO (1,2),NRESSW
      C
          1 CALL RESCAL(KF,KRF,KRC)
            RETURN
      C
          2 CALL RESCL1(KF,KRF,KRC)
            RETURN
            END
      C
      C
            SUBROUTINE RESCAL(KF,KRF,KRC)
      C     CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
      C     IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
      C     BEFORE INJECTION, THE RESIDUAL IS SCALED
      C     BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
      C     FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
      C     GRIDSIZE ON GRID KC.
      C      KRC     KC    KRF   KF  KF
      C     R   = 4*S   ( B   - A   U   )
      C               KF
      C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON Q(18000),IUF(200),IRF(200),IRC(200)
            CALL KEY(KF,IUF,IIF,JJF,HF)
            CALL KEY(KRF,IRF,IIF,JJF,HF)
            CALL KEY(KRC,IRC,IIC,JJC,HC)
            IIC1=IIC-1
            JJC1=JJC-1
            DO 1 IC=2,IIC1
            ICR=IRC(IC)
            IF=2*IC-1
            JF=1
            IFR=IRF(IF)
            IFO=IUF(IF)
            IFM=IUF(IF-1)
            IFP=IUF(IF+1)
            DO 1 JC=2,JJC1
            JF=JF+2
            S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
          1 Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
            RETURN
            END
      C
      C
```

AD-A096 652 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY--ETC(U)
OCT 80 A BRANDT, C W CRYER DAAG29-80-C-0041
UNCLASSIFIED MRC-TSR-2131 NL

2 OF 2
AD A096652

END
DATE FILMED
4-81
DTIC

```
      SUBROUTINE RESCL1(KF,KRF,KRC)
C     MODIFICATION #5 UPDATED JUNE 23 1980
C     USES WEIGHTED RESIDUALS NEAR THE BOUNDARY
C     CALCULATES THE RESIDUAL ON GRID KF WITH RIGHT HAND SIDE
C     IN ARRAY KRF , AND INJECTS INTO ARRAY KRC.
C     BEFORE INJECTION, THE RESIDUAL IS SCALED
C     BY MULTIPLYING BY THE FACTOR 4 TO TAKE ACCOUNT OF THE
C     FACT THAT THE GRID SIZE ON GRID KF IS HALF THE
C     GRIDSIZE ON GRID KC.
C      KRC     KC    KRF   KF  KF
C     R   = 4*S   ( B   - A   U   )
C              KF
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),IUF(200),IRF(200),IRC(200)
      DIMENSION R(9)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KRF,IRF,IIF,JJF,HF)
      CALL KEY(KRC,IRC,IIC,JJC,HC)
      IIC1=IIC-1
      JJC1=JJC-1
      DO 1 IC=2,IIC1
      ICR=IRC(IC)
      IF=2*IC-1
      JF=1
      IFR=IRF(IF)
      IFO=IUF(IF)
      IFM=IUF(IF-1)
      IFP=IUF(IF+1)
      DO 1 JC=2,JJC1
      JF=JF+2
      IF(Q(IFO+JF).EQ.0)GOTO 2
      IF(Q(IFP+JF+1).GT.0 .AND. Q(IFP+JF-1).GT.0 .AND.
     *   Q(IFO+JF+1).GT.0 .AND. Q(IFO+JF-1).GT.0 .AND.
     *   Q(IFM+JF+1).GT.0 .AND. Q(IFM+JF-1).GT.0 .AND.
     *   Q(IFM+JF  ).GT.0 .AND. Q(IFP+JF  ).GT.0 )GOTO 2
      N=0
      DO 3 I1=1,3
      I=IF+I1-2
      DO 3 J1=1,3
      J=JF+J1-2
      N=N+1
      IR=IRF(I)
      IO=IUF(I)
      IM=IUF(I-1)
      IP=IUF(I+1)
      S=Q(IO+J+1)+Q(IO+J-1)+Q(IM+J)+Q(IP+J)
      S=Q(IR+J)+4*Q(IO+J)-S
      IF(Q(IO+J).EQ.0)S=0
      R(N)=S
    3 CONTINUE
      Q(ICR+JC)=R(5)+.5*(R(2)+R(4)+R(6)+R(8)+
    1   .5*(R(1)+R(3)+R(7)+R(9) ))
      GOTO 1
    2 S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
      Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
    1 CONTINUE
```

```
      RETURN
      END
C
C
      SUBROUTINE SOLPRT(K,MPRINT)
C     NOT TIMED
C     PRINTS THE ARRAY K ON THE SUBARRAY MPRINT.
C     IF K<MPRINT, PRINTS ENTIRE ARRAY K
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),QTEM(100),IST(600)
      TIME=URTIMG(0)
      CALL KEY (MPRINT,IST,IIM,JJ,H)
      CALL KEY (K,IST,II,JJ,H)
      INTERV=1
      IF(K.GT.MPRINT)INTERV=(II-1)/(IIM-1)
      DO 20 J=JJ,1,-INTERV
      L=0
      DO 10 I=1,II,INTERV
C     X AND Y ARE NOT PRINTED HERE, BUT ARE COMPUTED IN
C     CASE A LATER VERSION NEEDS THEM.
      X=(I-1)*H
      Y=(J-1)*H
      L=L+1
      QTEM(L)=Q(IST(I)+J)
   10    CONTINUE
      PRINT *,(QTEM(LL),LL=1,L)
   20    CONTINUE
      CALL URTIMS(TIME)
      RETURN
      END
C
C
C
      SUBROUTINE SOLRED
C     NOT TIMED
C     PUTS ACCURATE SOLUTION INTO GRID NGRSOL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON Q(18000),ISTA(200),QTEM(600)
      COMMON /SOLTAU/M,NGRSOL,PT
      COMMON /PRBDAT/Y1,Y2,A,R
      COMMON /SWDAT/NFGSW,NINTSW,NRESSW
      TIME=URTIMG(0)
      CALL KEY(NGRSOL,ISTA,IIA,JJA,HA)
C
      GOTO(1,2),NFGSW
C
C     DAM PROBLEM
C     ACCURATE SOL IS DOUBLE PRECISION ON GRID M=7
C     WITH INITIAL GRID 2X3
C     STORED IN FILE 10.
    1 CONTINUE
      MA=7
      IIMA=2**(MA-1)*2+1
      INTERV=(IIMA-1)/(IIA-1)
      JJMA=(JJA-1)*INTERV+1
      REWIND 10
```

```
1027.                DO 20 JA=1,JJMA
1028.                READ(10) (QTEM(IA),IA=1,IIMA)
1029.                J=(JA-1)/INTERV+1
1030.                IF( (J-1)*INTERV .NE. JA-1 )GOTO 20
1031.                DO 10 I=1,IIA
1032.                IA=(I-1)*INTERV+1
1033.                Q(ISTA(I)+J)=QTEM(IA)
1034.             10 CONTINUE
1035.             20 CONTINUE
1036.                GOTO 1000
1037.          C
1038.          C
1039.          C     PROBLEM OF SECTION 5: (5.3) AND (5.4)
1040.          C     EXACT SOLUTION KNOWN
1041.              2 CONTINUE
1042.                D=2.5*R
1043.                DO 30 I=1,IIA
1044.                IO=ISTA(I)
1045.                DO 25 J=1,JJA
1046.                X=(I-1)*HA
1047.                Y=(J-1)*HA
1048.                A=DMAX1(0.D0,D-R*X-Y)
1049.                B=X+Y
1050.                G=A*A*(DCOS(B)+2)
1051.                Q(IO+J)=G
1052.             25 CONTINUE
1053.             30 CONTINUE
1054.                GOTO 1000
1055.          C
1056.           1000 CONTINUE
1057.                CALL URTIMS(TIME)
1058.                RETURN
1059.                END
1060.          C
1061.                SUBROUTINE SUBTRC(KF,KC)
1062.          C     THIS SUBROUTINE COMPUTES THE VALUE INJECTED FROM GRID KF TO
1063.          C     GRID KC AND SUBTRACTS IT FROM THE SOLUTION ON GRID KC.
1064.          C      KC     KC     KC  KF
1065.          C     W   = U    - I   U
1066.          C                    KF
1067.          C
1068.                IMPLICIT DOUBLE PRECISION (A-H,O-Z)
1069.                COMMON Q(18000),IUF(200),IUC(200)
1070.                CALL KEY(KF,IUF,IIF,JJF,HF)
1071.                CALL KEY(KC,IUC,IIC,JJC,HC)
1072.                DO 1 IC=1,IIC
1073.                IF=2*IC-1
1074.                IFO=IUF(IF)
1075.                ICO=IUC(IC)
1076.                JF=-1
1077.                DO 1 JC=1,JJC
1078.                JF=JF+2
1079.                Q(ICO+JC)=Q(ICO+JC)-Q(IFO+JF)
1080.              1 CONTINUE
1081.                RETURN
1082.                END
1083.          C
```

```
1084.      C
1085.            SUBROUTINE TAUCAM(KU,KR,KF,ITAU,TAUGNM)
1086.      C     COMPUTES TAU AND TAU-Z GREEN NORM
1087.      C     UPDATED AUGUST 26 1980
1088.      C     PERFORMS TAU EXTRAPOLATION IF ITAU=1
1089.      C       BY ADDING TAU TO RHS ON GRID
1090.      C     GRID KU CONTAINS U
1091.      C     GRID KR CONTAINS SUM OF FIRST TWO TERMS IN (6.7)
1092.      C          PREVIOUSLY OBTAINED USING RESSW AND CORSRE
1093.      C     GRID KF CONTAINS THIRD BRACKET IN (6.7) PREVIOUSLY
1094.      C          COMPUTED BY RESBW
1095.      C     ITAU IS PARAMETER WHICH DETERMINES WHETHER EXTRAPOLATION
1096.      C          WILL BE PERFORMED
1097.      C     TAUGNM IS RETURNED AS GREEN NORM OF TAU-Z
1098.      C
1099.      C      K-1    PT       K-1   K   K K        K-1 K-1 K        K-1 K
1100.      C     T    = 2    *(4S    (B  - A U )) + (A    I   U ) - (4S    B ) )
1101.      C           -------   K                          K            K
1102.      C           2**PT-1
1103.            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
1104.            COMMON Q(18000),IKR(200),IKF(200),IKU(200)
1105.            COMMON /SOLTAU/M,NGRSOL,PT
1106.            CALL KEY(KR,IKR,II,JJ,HK)
1107.            CALL KEY(KF,IKF,II,JJ,HK)
1108.            CALL KEY(KU,IKU,II,JJ,HK)
1109.            A=2.**PT/(2.**PT-1)
1110.            TAUGNM=0
1111.            II1=II-1
1112.            JJ1=JJ-1
1113.            DO 1 IK=2,II1
1114.            IRK=IKR(IK)
1115.            IFKO=IKF(IK)
1116.            IO=IKU(IK)
1117.            IM=IKU(IK-1)
1118.            IP=IKU(IK+1)
1119.            DO 1 JK=2,JJ1
1120.            T=Q(IRK+JK)-Q(IFKO+JK)
1121.            T=A*T
1122.            IF(Q(IO+JK).EQ.0)T=0
1123.            TAUGNM=TAUGNM+T*T
1124.              IF( Q(JK+IO+1).EQ.0 .OR.
1125.           * Q(IO+JK-1).EQ.0 .OR. Q(IM+JK).EQ.0 .OR.
1126.           * Q(IP+JK).EQ.0) T=0
1127.              IF(ITAU.EQ.1)Q(IRK+JK)=T+Q(IFKO+JK)
1128.           1  CONTINUE
1129.              TAUGNM=SQRT(TAUGNM)/HK
1130.              RETURN
1131.              END
```

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MRC-TSR-2131	2. GOVT ACCESSION NO. AD-A096 652	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Achi Brandt and Colin W. Cryer		8. CONTRACT OR GRANT NUMBER(s) DAAG29-80-C-0041 MCS77-26732
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit # 3 (Numerical Analysis and Computer Science)
11. CONTROLLING OFFICE NAME AND ADDRESS See Item 18 below.		12. REPORT DATE October 1980 13. NUMBER OF PAGES 96
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Technical Summary Rept.		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

National Science Foundation
Washington, D.C. 20550

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Multigrid Algorithms
Free Boundary Problems
Linear Complementarity Problems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We show that the multigrid algorithms of Brandt can be adapted to solve linear complementarity problems arising from free boundary problems. The multigrid algorithms are significantly faster than previous algorithms. Using the multigrid algorithms, which are simple modifications of multigrid algorithms for equalities, it is possible to solve the difference equations to within truncation error using less work than the equivalent of six Gauss-Seidel sweeps on the finest grid.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 222200

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)